НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Автоматики та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК 004.8_____

«До захисту допущено»
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
"04" грудня 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації)___121 інженерія програмного забезпечення _
(код і назва спеціальності)

_____

на тему: Інтелектуальна ігрова система з використанням нейронної мережі_____

Виконав (-ла): студент (-ка) 6 курсу, групи ІТ-73мп
(шифр групи)

_____Ліхоузов Сергій Олексійович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник _____к.т.н., доцент, Дорогий Я.Ю_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

АНОТАЦІЯ

Магістерська дисертація містить 100 сторінок, 8 рисунків, 31 таблиць, список літератури з 25 найменувань та 9 додатків.

Ключові слова: інтелектуальна система, нейронна мережа, формалізація.

Метою магістерської дисертації є проектування та розробка інтелектуальної ігрової системи, що реалізує концепцію гри “Морський бій”.

Об’єктом дослідження є інтелектуальні ігрові системи.

Предметом дослідження є нейронна мережа.

В результаті дослідження було створено застосування з використанням нейронної мережі.

ANNOTATION

The master's dissertation contains 100 pages, 8 figures, 31 tables, a list of literature of 25 titles and 9 applications.

Key words: intellectual system, neural network, formalisation.

The purpose of the master's thesis is to design and develop an intellectual game system that implements the concept of the "Sea Battle" game.

The object of research is the intellectual game systems

The subject of research is the neural network.

As a result of the study application was created using the neural network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ВВ – варіант використання

ДВВ – діаграма варіантів використання

ДП – діаграма прецедентів

ДКА – діаграма кінцевих автоматів

ДС – діаграма станів

КА – кінцевий автомат

ПЗ – програмне забезпечення

FSM – finite state machine

UML – unified modeling language

ВСТУП

Актуальність роботи

Актуальність роботи полягає в тому, що на даний момент ключовою характеристикою будь-якої комп’ютерної гри є достовірність вражень, які сприймає гравець, а достовірність досягається реалістичними діями штучного інтелекту, який задіяний у абсолютній більшості ігро. Ішні характеристики, такі як графічний супровід та аудіо супровід неможливо зробити кращими за конкурентів при невеликому або навіть середньому бюджеті тому, що на розробку головних конкурентів витрачаються сотні мільйонів доларів. Єдиний шанс бути кращим за конкурентів це зробити гру, яка буде достовірною та цікавою, на що також впливає ефективність гри, яку задає штучний інтелект.[1]

Штучний інтелект впливає на всі аспекти гри, створюючи саме такі умови, які необхідні для того, щоб зацікавити гравця і він продовжував грати.

Будь-які характеристики діяльності гравця, наприклад, час прийняття рішення, швидкість реакції, тип обраної стратегії та періодичність ігрових сеансів можуть стати параметрами навколишнього середовища ігрової системи, тобто складність гри.

Але можливості штучного інтелекту довзоляють змоделювати протилежну ситуацію: вже керований штучним інтелектом агент (персонаж) буде виконувати дії гравця, що довзоляє симулювати виконання осмисленних дій, групове переміщення, переслідування, ухиляння та пошук шляху, а в загальному випадку – будь-які дії разом чи окремо, повністю симулюючи дії живої людини.

Найбільш часто використовуваним способом застосування штучного інтелекту є реалізація системи, здатної прийматі ігрові рішення або симуляція цілої системи з метою досягти необхідного рівня реалістичності внутрішньо ігрової поведінки ігровими агентами.

“Грати – приймати ігрові рішення, маючи ту ігрову інформацію, яка по

правилам та по ходу гри доступна для гравця. Приймати ігрові рішення можливо не користуючись вхідними даними, проте у більшості ігор це не призводить до перемоги. Отже, гравець робить наступні кроки на основі попередніх. Можна сказати, що гравець на основі параметрів продукує результат у вигляді первних дій. За прийняття рішення на основі вхідних параметрів відповідає штучний інтелект.”[1]

Об’єктом дослідження даної роботи є інтелектуальні ігрові системи.

Метою магістерської дисертації є проектування та розробка інтелектуальної ігрової системи, що реалізує концепцію гри “Морський бій” з використанням технологій Python та TensorFlow.

Об’єктом дослідження є інтелектуальні ігрові системи.

Предметом дослідження є нейронні мережі

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Різновиди штучного інтелекту в іграх

Всі види та реалізації штучного інтелекту розділено на три типи за основною ознакою – обробкою параметрів, які штучний інтелект отримує на вхід:

- тип з завчасно прописаними діями;
- реагуючий тип;
- тип машинного начання.

У кожний з цих типів входять окремі підходи, які буде розглянуто окремо. Такий розподіл дозволяє розділити види штучного інтелекту по тому, як вони реагують на вхідні параметри. У кожну з цих груп входять декілька методів.

Було обрано три критерії для оцінки та вибору кращого типу:

- складність розробки;
- реалістичність поведінки агентів під управлінням штучного інтелекту;
- масштабованість [1].

Є й інша класифікація, в якій розділення проходить по наявності та можливості застосовувати пам’ять про попередні дії штучного інтелекту. Це розділення має обмеження в тому, що воно досить нечітке тому, що буває важко розділити кінець однієї дії та початок іншої, а також не враховується існування штучного інтелекту, який принципово ігнорує не тільки попередні дії, а й зміни в стані навколишнього середовища тому, що, наприклад, його дії повністю випадкові або ігрове середовище не змінюється або змінюється дуже мало, тому можливо прописати всі дії для деякого періоду часу та зациклити їх при потребі.

На даний момент є такі типи ШІ:

- Завчасно прописані дії, механізм яких описаний вище. Не змінює своєї поведінки в залежності від обставин в будь-якому разі тому. Складно реалізуємий у випадку динамічно змінного навколишнього ігрового середовища.
- “Реагуючий тип. Даний тип характеризується наявністю правил поведінки, які визначають поведінку керованого штучним інтелектом реагуючого типу об’єкту. Параметри навколишнього середовища, які змінюються, або не змінюються, інтерпретуються та на основі прийнятого рішення виконуються деякі дії. Основною особливістю є те, що дані правила, по яким приймаються рішення, не змінюються чи змінюються зовні (наприклад, розробник змінив правила реагування на деякий чинний), проте ніколи не змінюються зсередини.[1]
- Машинне навчання. Цей тип штучного інтелекту не так просто реалізувати навіть для виконання простих завдать, проте він легко машстабується для виконання задач у змінному середовищі якраз за допомогою самонавчання (навчання з вчителем, без вчителя, з підкріпленням та ін.). Це надає даному типові штучного інтелекту гнучкість, ящої немає у інших типів.

1.1.1 Типові завдання для штучного інтелекту в іграх

Пошук шляху

“Існує безліч різних проблем, пов'язаних з пошуком шляхів. На жаль, жодне рішення не підходить для кожного типу проблеми пошуку шляху. Рішення залежить від специфіки вимог до пошуку шляху для будь-якої заданої гри. Наприклад, задачу сильно змінюють такі факти, як можливість руху цілі, наявність перешкод, вплив ландшафту або його наявність, оптимальність найкоротшого шляху, а також факт того, що проблема пошуку може навість у відповіді вимагати досягнення певного місця призначення. Можливо, завдання полягає в тому, щоб ігровий персонаж міг рухатися або досліджувати ігрове середовище розумно.

На самому базовому рівні пошук шляхів – це просто процес переміщення позиції ігрового персонажа з його первинного місця розташування до бажаного місця призначення.”[1]

Випадкове переміщення може бути простим і ефективним методом запобігання перешкод. Особливо гостро це працює в навколишнє середовище з відносно невеликою кількістю перешкод. Ігрове середовище з рідко розміщеними деревами є хорошим кандидатом для методу випадкового переміщення.

Існуючий можливий спосіб реалізації штучного інтелекту з завчасно прописаними діями – це опис програмними способами шляхом прописування дій по виконанню алгоритму, описаному гейм дизайнером – розробником ігрового середовища. Даний алгоритм потрібно буде змінювати кожен раз, коли необхідно змінити будь-який параметр ігрового середовища, що впливає на можливість логічної реалізації дій алгоритму.

Штучний інтелект реагуючого типу дає можливість реалізувати свою логіку декількома способами. Відстеження перешкод – ще один відносно простий спосіб запобігання перешкод. Цей метод може бути використаний, коли ви намагаєтеся знайти шлях навколо великих перешкод, таких як

гірський хребет в стратегії або ролі, граючи. За допомогою цього методу керований комп'ютером символ слід простому алгоритму пошуку шляху спробувати досягти своєї мети. Він триває з його шляху, поки не досягне перешкоди. У цей момент він переключається на відстеження стану. У стані трасування він слід за краєм перешкоди, намагаючись обійти його.

Метод хлібних крихток може змусити персонажів, керованих комп'ютером, бути дуже розумними, тому що гравець неусвідомлено створює шлях для керованого комп'ютером персонажу. Кожен раз, коли гравець робить крок, він

неусвідомлено залишає невидимий маркер або хлібну крихту в ігровому світі. Коли ігровий персонаж приходить контакт з цим місцем, він просто починає йти по слідах. Персонаж гри буде слідувати по стопах гравця до досягнення гравця. Складність шляху і кількість перешкод у шлях не має значення. Гравець вже створив шлях, тому серйозних обчислень не потрібно.

Метод хлібних крихток також є ефективним і ефективним способом переміщення груп контрольованих комп'ютером персонажів. Замість того, щоб кожен член групи використовував дорогу і трудомістку перевірку шляху

алгоритм, ви можете просто кожен член слідувати за панібрамі лідера.

В реальній грі кількість крихток буде залежати від конкретної гри та наскільки складний потрібен штучний інтелект, щоб з'являлися ігрові персонажі.

Тип штучного інтелект машинне навчання довзоляє реалізовувати рішення проблеми пошуку шляху відносно малими зусиллями, проте для деяких випадків реалізація іншими способами призведе до меншої витрати часу.

Імітація осмисленних дій

“Часто в відеоіграх персонажі повинні переміщатися у скупчених групах, а не самостійно. Розглянемо деякі приклади. Для створення гри необхідно створити недалеко від головного міста луг та овець, що пасуться на ньому. Вівці виглядали б реалістичніше, якби вони паслися в отарі, а не

гуляли навкруги безцільно. Можливо, в цій же грі є зграя птахів, які полюють на ігрових персонажів-жителей. І тут птахи, які полюють на стада, а не літають безцільно, виглядають реалістичнішими і кидають виклик гравцеві, створюючи завдання по протистоянню деяким чином взаємодіючим групам хижаків. За допомогою такого прикладу неважко побачити, що таку саму логіку можна застосувати для моделювання поведінки при спільному пересуванні велетенських мурашок, бджіл, щурів або морських істот. З урахуванням вищезазначеного, можливо розповсюдити застосування спільнго пересування до будь-яких створінь не під контролем гравця. Наприклад, у стратегіях в режимі реального часу можливе використання для моделювання переміщень створінь під контролем гравця. У симуляції бойового польоту ви можете застосувати такий груповий рух до комп'ютерних ескадронів літаків. У шутері від першої особи, керовані комп'ютером ворожі або дружні загони можуть використати такий груповий рух. Можливе навіть використання варіацій базової поведінки групового руху, наприклад, імітуючи натовпи людей, що ходять навколо міської площі.”[1]

Скоординований груповий рух, а точніше його видимість, виникає при реалістичному виконанні реалізації алгоритмів, що втілюють імітацію осмисленних дій.

Для штучного інтелекту з завчасно прописаними діями цілком можливо повністю прописати пересування та дії для всіх, хто бере участь у імітації осмисленних дій, проте складність такої імітація підвищується у стільки разів, скільки учасників руху потрібно прописати.

Для штучного інтелекту реагуючого типу є декілька способів зробити імітацію. Один із них описаний Крейгом Рейнольдсом у статті “Стаї, стада та школи: розподілена модель поведінки”. В оригінальній формі алгоритм представлений для імітації зграй птахів, риб або інших істот.

Для штучного інтелекту з можливістю навчання складність задачі не відрізняється від задачі пошуку шляху тому, що в тому й іншому випадках

потрібно зарпограмувати систему, яка сама буде навчатися виконувати дії, і різниця може бути лише у швидкості навчання, проте на час розробки це не впливає.

Переслідування та ухиляння

“Незалежно від типу гри, майже завжди необхідно реалізувати механізм передбачення руху цілі та механізм прорахунку оптимального маршруту.

У екшні або в аркадній грі ситуація може включати ворожі космічні кораблі і задіяти корабель гравця. У пригодницькій рольовій грі може включати наявність троля або іншої прекрасної істоти, переслідуючого гравця. У шутерах від першої особи і симуляціях польоту, можливо, доведеться направляти керовані ракети і знищувати гравця або його літак. У будь-якому разі вам потрібна деяка логіка, яка дозволяє хижакам, керованим штучним інтелектом, наздоганяти здобич, а тій, в свою чергу, збільшувати відстань між собою та переслідувачем.”[1]

Для того, щоб реалізувати сам процесс переслідування (ухиляння), необхідно враховувати можливість наявності перешкод на шляху як втікача, так і того, хто наздоганяє втікача. Перешкоди можуть бути спільними або для кожного із учасників вони можуть бути своїми, проте частіше всього вони спільні. В більшості випадків (в тих, в яких перешкоди спільні), процесс оминання перешкод можна спростити до методу хлібних крихток, коли той, хто наздоганяє, просто повторює траєкторію того, хто втікає.

Для штучного інтелекту із завчасно прописаними діями дуже складно реалізувати переслідування або ухиляння саме тому, що він не враховує положення антагоніста. Можливість реалізувати реалістичну поведінку є тільки у випадку, коли шлях для переслідування існує лише один (наприклад, стежка в лісі без перехресть). Якщо ж прописати випадковий напрям руху, результат буде поганим – керований штучним інтелектом персонаж в деяких випадках буде рухатися у протилежному до досягнення мети напрямі.

Для штучного інтелекту реагуючого типу знову існує декілька способів виконати це завдання.

Найпростіший та легкий у розробці спосіб, який можна використати для того, щоб реалізувати переслідування включає в себе оновлення координат “хижака” кожен ігровий цикл таким чином, щоб різниця між його координатами та координатами цілі ставала меншою. Цей алгоритм не зважає та перешкоди, що може влпинути на реалістичність, проте це найпростіший та найшвидший у застосування алгоритм.

На додачу до цього методу існують інші, які краще задовольняють потребу у реалізації переслідування чи ухиляння.

Наприклад, використання потенційних функцій, які буде описано нижче, або розрахунок можливого положення антагоністу для того, щоб реалізувати перехоплення цілі або превентивне збільшення відстані у випадку ухиляння.

Для інтелекту з можливістю навчання існують декілька можливостей для реалізації рішення. Наприклад, генетичний алгоритм дозволить вивести покоління утікачів, які будуть все краще ухилятися, або переслідувачів, які будуть еволюціонувати так само, як це робили та роблять хижаки в реальному світі. Також існують інші способи.

Обирання стратегії

“Обирання стратегії – узагальнююче поняття для дій, яке включає в себе одномоментний вибір з декількох альтернатив, таких як, наприклад, оборона чи наступ, вибір зброї ближнього чи дальнього бою тощо. Стратегія – концепція, яку неможливо реалізувати на першому рівні тому, що в ньому прописані усі дії, або в між ними немає зв’язку, який можна було б охарактеризувати як стратегію. На рівні реагування стратегії можно можливо зробити за допомогою дерев рішень, для яких розробник прописує всю можливу логіку та приймає рішення, які з вхідних параметрів доцільно включити у логіку дерева рішень. У окремих випадках обирати потрібно в межах однієї дії (залежить від типу гри).”[1]

Прикладом може бути майже кожна дія (це залежить від рівня абстракції, на якому розглядається дія у грі) – обирання зброї, обирання виду дій (наступ чи оборона) та обирання низки дій – розвідувальної тактики чи

зневажання на дії опонента з метою максимального розвитку. Рисунок 1.1 демонструє приклад оформлення логіки обирання стратегії у шутері.

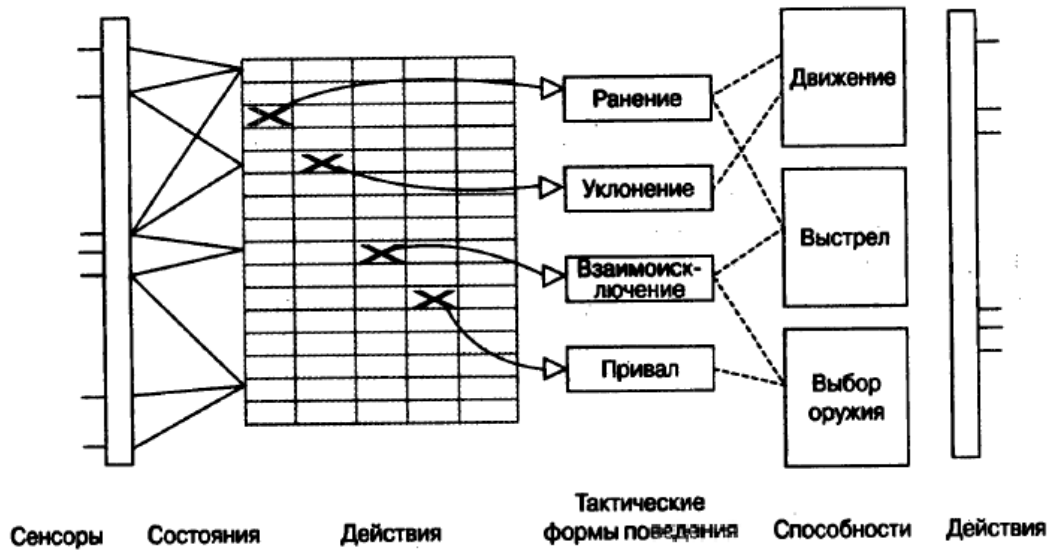

Рисунок 1.1 – приклад логіки обирання стратегії на прикладі шутеру[1]

“Обирання стратегії – яскравий приклад задачі, у якій перевага машинного навчання стає очевидною тому, що штучний інтелект сам формує стратегію та, завдяки більшій кількості часу, ніж доступно розробникові системи, робить це якісніше. Більше того, система може знайти стратегії там, де звичайна людина робила би безсистемні спроби в пошуках закономірностей.”[1]

Отже, розглянувши всі типи штучного інтелекту, можна звести отримані дані у таблицю та порівняти дані типи (Таблица 1.1).

Таблиця 1.1 – порівняння типів штучного інтелекту

	Перший тип	Другий тип	Третій тип
Реалістичність поведінки програмних агентів, які втілюють алгоритми штучного інтелекту	Достатній рівень	Достатній рівень	Повна реалістичність або будь-який необхідний заданий результат
Масштабування програмного рішення	Неможливе	Можливе, проте не у всіх випадках	Можливе
Витрати на розробку програмного рішення	Відрізняються від дуже низьких до дуже високих	Відрізняються від середніх до дуже високих	Відрізняються від середніх до високих

Отже, враховуючи вищеперераховані переваги типу штучного інтелекту машинного навчання, його буде обрано для реалізації інтелектуальної ігрової системи.

1.2 Існуючі методи машинного навчання

Нейронні мережі

Нейронні мережі набули широкого розповсюдження в останні роки і тепер сфери їх застосувань перелічюються десятками, а у найближчому майбутньому будуть перелічуватися сотнями. Причина такого швидкого розповсюдження в тому, що вони дозволяють автоматизувати ті сфери, які завжди потребували людської праці, крім того, нейронні мережі поряються в деяких випадках краще, крім того, у розвитку мереж є великий потенціал тому, що кількість нових розробок та відкриттів тільки росте з часом, як і практична користь від них – від раннього діагностування раку до новітніх систем безпеки, від вигадування рецептів до інновацій в сфері розваг. Поєднавши нейронні мережі та сферу компьютерних ігор, можна створити дещо одночасно і корисне у практичному сенсі, так і актуальне у науковому.

Нейронна мережа складається з нейронів, які можуть бути різних типів, і які оброблюють інформацію, яка надходить до них по каналах зв’язку, які також визначаються при проектуванні системи. Видів нейронних мереж дуже багати, принципово вони відрізняються:

- Наявністю прихованих шарів нейронів, тобто тих нейронів, результат обробки інформації якими не видно;
- Кількістю прихованих шарів, що впливає на можливість розшифрувати шлях, яким дана нейронна мережа дала саме такий результат;
- Видом нейронів, що впливає на спосіб обробки (або зберігання) інформації;
- Типами зв’яку між нейронами;
- Наявністю чи відсутністю зворотнього зв’язку у нейронній мережі;
- Кількістю нейронів у шарі.

Генетичні алгоритми

Генетичний алгоритм – алгоритм поступового підбору, рекомбінування і ітерації деяких змінних.

Генетичні алгоритми відрізняються від всіх інших алгоритмів тим, що вони використовують змішування при ітерації, що означає, що на основі більш успіхних варінтів поточної ітерації буде обрано кращі та на їх основі побудовано нові ітерації, що призводить до появи деякої варіативності та можливості досягти кращих результатів. Основним ресурсом для генетичних алгоритмів є час тому, що за достатьо великий період зазвичай можливо знайти відповідаючи умовам задачі рішення.

Таким критерієм, який задовольняє умови задачі, в більшості випадків може бути:

- знаходження глобального, або надоптимального рішення деякої задачі;
- проходження деякої кількості ітерацій;
- вичерпання часу, відпущеного на покращення результатів, без зважання на кількість ітерацій.

Недоліком, який не дозволяє використовувати генетичні алгоритми у навчанні є те, що цей метод потребує обробки всіх можливих комбінацій, а їх загальна калькість відносно кількості потенційно корисних дій у випадку гри дуже велика, тобто у цьому випадку процесс навчання буде дуже довгим.

Байесова мережа

Байєсова мережа – це модель. Вона відображає стан певної частини світу, який моделюється, і описує, як ці стани пов'язані з ймовірністю. Модель може бути вашим будинком, вашим автомобілем, вашим тілом, вашою громадою, екосистемою, фондовою біржею тощо. Абсолютно все, що може бути моделюється мережею Байєса. Всі можливі стани моделі відображають всі можливі існуючі світи, тобто всі можливі шляхи, якими можна налаштовувати частини або стани. Двигун автомобіля може нормально працювати або створювати проблеми. Це шини можуть бути надутими або плоскими. Ваше тіло може бути хворим або здоровим тощо.

Деякі стани, як правило, трапляються частіше, коли присутні інші держави. Таким чином, якщо ви хворієте, шанси на те, що вологість, є вищими. Якщо він хмарний, то шанси на дощ вище, і так далі.

Ось проста мережа Байєса, яка ілюструє ці поняття. У цьому простому світі, скажімо, погода може мати три стани: сонячна, хмарність або дощова, також те, що трава може бути мокрим або сухим, а також можна включити або виключити спринклер. Тепер у цьому світі існують деякі причинні зв'язки. Якщо воно дощове, то воно полегшить траву прямо. Але якщо він довгий час сонячний, це теж може зробити траву мокрим, побічно, змусивши нас включити спринклер.

Байесові мережі використовуються в таких напрямах:

- Менеджмент фінансових ризиків
- Моделювання екосистем
- Передбачення
- Діагностика

Обмеження байесової мережі:

- складність обчислення;
- байесова мережа може викликати складнощі у випадку наявності великої кількості випадкових параметрів тому, що намагається поставити їх у причинно-наслідкові зв’язки. Внаслідок цього

справжні зв’язки втрачаються або втрачають значущість. Байєсова мережа кодує лише спрямовані зв'язки, а не двонаправлені. Байесова мережа не дає жодних гарантій щодо зображення причинно-наслідкових зв'язків[3].

- виходячи з попереднього пункту, якщо дані були згенеровані з моделі, в якій щонайменше три змінні співвідносяться між собою, байесова мережа не зможе змоделювати ці зв’язки.
- одним з найважливіших проблем з BN є те, що деякі зі складних функцій забивання потребують надійних приорітетів, щоб знайти структуру, яка ближче до оригінальної моделі.

Зважаючи на наявні кращі альтернативи та серйозні обмеження, а також складність розробки системи з використанням байесової мережі, було прийнято рішення відмовитися від розробки інтелектуальної ігрової системи з використанням байесової мережі.

Навчання з підкріпленням

Навчання з підкріпленням – область машинного навчання, пов'язана з тим, як програмні агенти повинні робити дії в середовищі, щоб досягти деякого результату, який позначено як найкращий. Ця проблема, в силу своєї спільності, вивчається у багатьох інших дисциплінах, таких як теорія ігор, теорія управління, дослідження операцій, теорія інформації, оптимізація на основі моделювання, статистика і генетичні алгоритми. У літературі подослідженнях і управлінні операціями навчання підкріпленню називається наближеним динамічним програмуванням або нейродинамічним програмуванням. Проблеми, що цікавлять, в навчанні з підкріпленням також вивчалися в теорії оптимального управління, яка в основному пов'язана з існуванням і характеристикою оптимальних рішень, а також з алгоритмами для їх точного обчислення і у меншій мірі з навчанням або наближенням, особливо у відсутність математична модель середовища. У економіці і теорії ігор навчання з підкріпленням може використовуватися для пояснення того, як рівновага може виникати при обмеженій раціональності. У машинному

навчанні середовище зазвичай формулюється як процес ухвалення рішень Маркова (MDP), бо існує велика множина способів реалізації навчання з підкріпленням з методами динамічного програмування. Основна відмінність між класичними методами динамічного програмування і алгоритмами навчання з підкріпленням полягає в тому, що останні не припускають знання точної математичної моделі MDP і націлені навеликі MDP, де точні методи стають неможливими.

Навчання з підкріпленням вважається однією з трьох парадигм машинного навчання, разом з навчанням під спостереженням і навчанням без учителя. Він відрізняється від контрольованого навчання тим, що правильні пари введення / виведення не вимагається, і неоптимальні дії не вимагають явного коригування. Замість цього основна увага приділяється продуктивності, яка включає пошук балансу між розвідкою (недослідженій території) і експлуатацією (з урахуванням сучасних знань). Компроміс між розвідкою і експлуатацією був найретельніше вивчений за допомогою проблеми багаторукого бандита і кінцевих MDP[4].

Навчання без втручання вчителя

Навчання без вчителя є одним із методів машинного навчання, яке реалізує ідею можливості навчання без зовнішнього втручання. Сутністю даної методики є припущення про те, що існують деякі приховані закономірності, які неможливо або дуже важко знайти і використати в навчанні з вчителем чи підкріпленням, тобто будь-яке зовнішнє втручання погіршить результат. Такий спосіб навчання можливий у випадках, коли правила існування системи, в якій необхідно знайти рішення деякої задачі, відомі, про неможливо знайти хоча б якість логічні зв’язки між окремими компонентами системи, тобто правила гри відомі, а стратегії ще ні.

Кластерний аналіз

Кластерний аналіз – задача розбиття заданої множини деяких об'єктів на підмножини так, щоб кожна підмножина складалася з найбільш схожими групами параметрів і відрязналася від інших підмножин. Кластерний аналіз це один із прикладів застосування методу машинного навчання без вчителя.

Кластерний аналіз – задача, для розв'язання якої використовуються різні підходи. Відносно того, що можна вважати кластером, а зо ні, алгоритм кластеризації може бути побудованим різними способами. Для того, щоб пояснити логіку реалізація кластерного аналізу, візьмемо приклад з задачею розділення країн на групи по характеристикам, наприклад, кількістю людей з вищим рівнем освіти, рівнем праці, етапом індустріального розвитку. Виконавши кластерний аналіз, отримаємо декілька підмножин загальної множини країн, і в одній із цих підмножин будуть Японія, Німеччина, Франція, Англія. В той час Уганда, Киргистан та Сомалі будуть в іншій підмножині тому, що вони поширюють між собою інший набір характеристик включаючи низький рівень життя, нестабільні та недемократичні інститути правління та низький рівень технологічного розвитку[5].

Навчання з вчителем

Навчання з учителем, або контрольоване навчання є один із варіантів машинного навчання, в ході якого система проходить ітерації за допомогою наявної множини прикладів пар вхідних параметрів та реакцій на них з метою визначення «реакції» для тих самих вхідних параметрів, які не належать наявній множини прикладів.

Між входами та результатами обробки може існувати (а може і не існувати) деяка залежність. У всіх випадках існує навчальна вибірка, в якій наявні обидва компоненти. На основі цих даних створюється залежність (побудова моделі відносин стимул-реакція, придатних для прогнозування). Для вимірювання точності відповідей, так само як і в навчанні на прикладах, може вводитися функціонал якості.

Глибоке навчання

Глибоке навчання – це техніка машинного навчання, а точніше сукупність технік машинного навчання, яка навчає комп'ютер робити те, що природно приходить людям: навчитися на прикладі.

При глибокому вивченні комп'ютерна модель навчається виконувати завдання класифікації безпосередньо з зображення, тексту або звуку. Моделі глибокого навчання можуть досягти найсучаснішої точності, іноді перевищуючи продуктивність людини. Моделі навчаються за допомогою великого набору помічених даних та архітектур нейронних мереж, що містять багато шарів.

Декілька прикладів застосування машинного навчання

- віртуальні ассистенти. Сірі, Кортана та Алекса використовують глибоке навчання для того, щоб навчитися розуміти мову людей;
- переклад. У схожий спосіб, алгоритми глибоко навчання можуть автоматично перекладати декілька мов. Це використовується та буде мати попит по всьому світі тому, що існують безліч людей, які не розуміють інших мов;
- чатботи та автовідповідачі. Вони використовуються для комунікації з клієнтами та не потребують людських ресурсів, тобто можуть майже безкоштовно замінити людей;
- додання кольору на фотографії. На даний момент існують безліч фотографій часів чорно-білої фотографії, які можливо перевести в колів за допомогою глибокого навчання;
- розпізнавання облич. Має попит у сфері безпеки та обслуговування, дозволяючи зменшити витрати на касирів (покупець буде здійснювати покупки за допомогою розпізнавання свого обличча);
- медицина та фармацевтика. Глибоке навчання дозволяє розпізнавати хвороби (наприклад, розпізнавання віку та рак за фотографію очей)

- персоналізація процесу покупок та дозвілля. Алгоритми глибокого навчання дозволяють персоналізувати асортимент товарів, які надаються клієнтові, що збільшує імовірність покупки або придбання послуги. Наприклад, Netflix використовує глибоке навчання для впровадження системи, яка надає глядачам фільми на спробу та має декілька сотень тисяч категорій глядачів, що дозволяє дуже ретельно підбирати товар для кожного клієнта.

Структура моделей глибокого навчання складається з нейронної мережі з декількох шарів, з яких деякі – приховані, тобто інформація обробляється декілька разів всередині них. Одним із популярних видів нейронних мереж в такому випадку стають конволюційні мережі, приклад такої наведено на рисунку 2.1

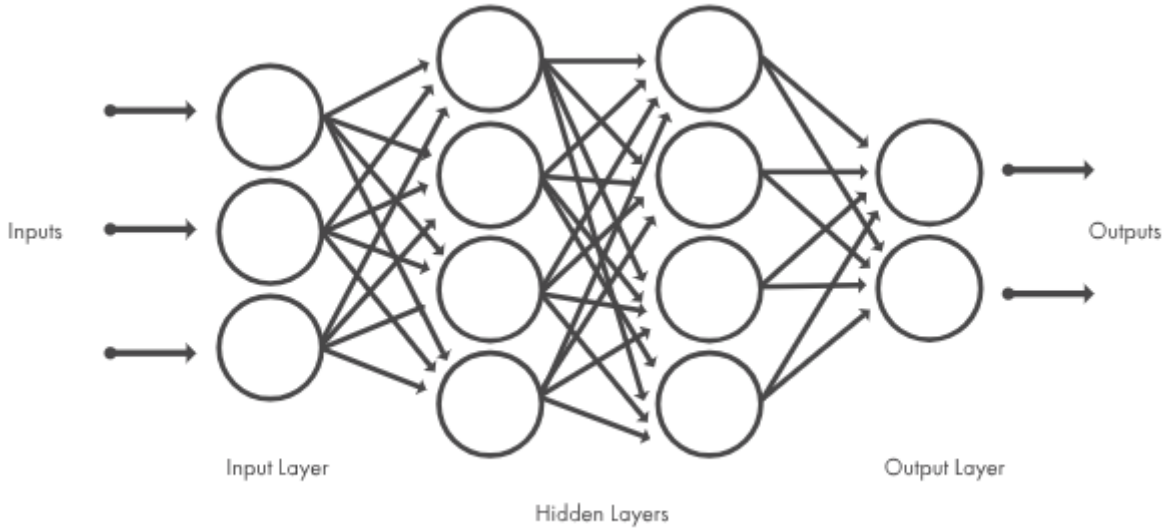

Рисунок 2.1 – загальна модель конволюційної нейронної мережі

Градієнтний спуск

Градієнтний спуск – це повторно застосовуваний алгоритм знаходження екстремуму дійсної функції першого порядку. Якщо натомість здійснюються кроки пропорційно самому значенню градієнту, то відбувається наближення до локального максимуму цієї функції; і ця процедура тоді відома як градієнтний підйом. Він може застосовуватися у будь-якій кількості вимірів. Може бути поєднаним разом із таким видом пошуку, як лінійним, що здійснює пошук локальном оптимальний розмір кроку, проте це може бути неефективним у плані часу[6].

Градієнтний спуск відомий також як найшвидший спуск, або метод найшвидшого спуску.

Обмеженнями застосування градієнтного спуску є те, що в деяких задачах його асимптотичний спосіб є ненайеффективнішим.

Висновки щодо вибору методу машинного навчання

Техніки навчання з підкрепленням можуть бути застосованими для навчання алгоритмів з метою досягти ефективного прийняття рішень. В той час, коли навчання з вчителем тренуються лише на даних, які вже доступні, навчання з підкріпленням продовжує покращувати результати алгоритму під час отримання інформації. Отже, від навчання з вчителем навчання з підкріпленням відрізняється можливістю навчатися під час процесу гри, що покращує результати в реальному часі. Крім того, навчання з підкріпленням надає можливість оминати навчання на нереалістичних ситуаціях, що призводить до економії часу навчання та дозволяє концентруватися лише на тих прикладах, які дозволять покращити результат саме для тої вибірки (наприклад, покращити результат зі схемою розстановки того гравця, який використовує систему), яка максимально схожа на наступні.

Висновки до розділу

Отже, порівнюючи типи штучного інтелекту, бачимо перевагу типу машинного навчання над типом с прописаними діями у деяких випадках у складності розробки, а також абсолютну перевагу у масштабованості, а також у реалістичності можливість отримати кращий результат, якщо це потрібно (проте не завжди є потреба)

Порівнюючи тип машинного навчання з реагуючим типов, машинне навчання перемагає по критерію складності розробки та масштабованості, а також, у деяких випадках, він виявляється кращим у реалістичності (коли це потрібно).

З видів машинного навчання обираємо тип глибоко навчання, в якому буде використовуватись нейронна мережа, підкріплення та градієнтний спуск.

2 ТЕОРЕТИЧИНИЙ ОПИС РІШЕННЯ

Алгоритми глибокого RL з політичним градієнтом складаються з трьох основних компонентів: управляюча мережа, градієнтного спуску і функції винагород. Їх буде деталізовано нижче і опишемо, як вони працюють разом.

2.1. Управляюча мережа

Політика для цього глибокого алгоритму навчання з підкріпленням – це нейронна мережа, яка відображає значення стану s у вірогідності для заданих ігрових дій a. Іншими словами, вхідний шар мережі приймає числове кодування середовища – стан гри в деякий ігровий момент. Коли цей вхід подається через мережу, значення на вихідному рівні відповідають логарифмічній вірогідності того, що кожна з доступних дій є оптимальною – один вихідний вузол є присутнім для кожної можливої дії, яку на даний ігровий момент можна обрати гравцеві. Отже, якщо тільки одна дія буде доступною на деяких момент, то вірогідність пострілу туди повинна бути максимальною. Проте, якщо наша мережа не знає, яка дія є оптимальною, більш ніж один вихідний вузол матиме кінцеву вагу.

Щоб проілюструвати вищевикладене, ми представляємо схему мережі, використану нижче. Для простоти розглянемо випадок з одним кораблем. Потім ми кодуємо наші поточні знання про стан ігрової дошки, взявши один вхідний нейрон для кожної позиції сітки нашого опонента. Зокрема, було запропоновано наступний вид запису кожного нейрона:

$$x_{0,i} = \begin{cases} -1, \text{не стріляв по } i\ ; \\ 0, \text{стріляв по і, корабля немає}\ ; \\ +1, \text{стріляв по і, корабель наявний}\ . \end{cases}$$

На рис 2.1 наявні п’ять вхідних нейронів, тому розмір дошки також п’ять. Саме в цьому полягає масштабованість підходу з машинним навчанням – змінивши лише один параметр, можливо створити систему, яка

буде грати оптимально для дошки будь-якого розміру. Перші три нейрони мають значення -1, що означає що ці клітинки ще не перевірялися на наявність корабля. Останні дві мають значення, відповідно, +1 та 0, що означає що корабель наявний на четвертій клітинці, проте не на п’ятій.

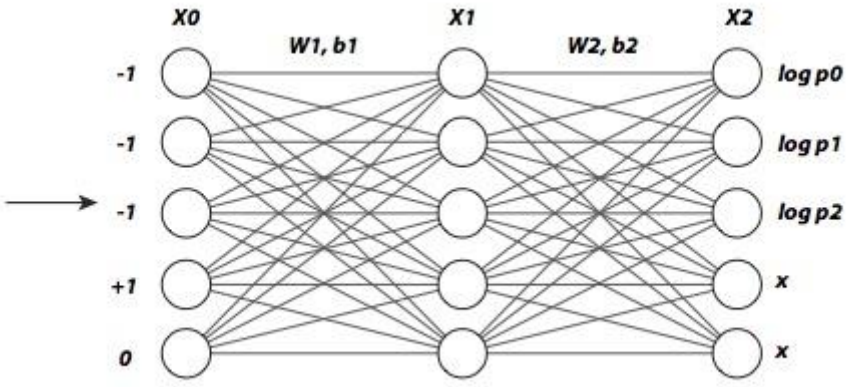

Рисунок 2.1 – функція, яка перетворює наявний стан грального дошки у наступну дію

Перші три значення у шарі вихідних значень нейронної мережі позначені логарифмами ймовірності. Ці значення відповідають ймовірності того що саме в цю клітинку потрібно робити наступний постріл. Неможливо зробити ще один постріл на четверту та п’яту позиції, тому їх потрібно ігнорувати, навіть якщо нейронна мережа робить якісь прогнози щодо них.

2.2 Функція винагороди

Для того, щоб тренувати алгоритм навчання з підкріпленням, потрібно побудувати процесс ітеративного навчання – алгоритм намагається розв’язати задачу, обираючи ті чи інші ходи, залежно від ймовірностей, які йому надає нейронна мережа. Якщо ходи були успішними, потрібно збільшити ймовірність цих дій в подальшому.

Функція винагороди це інструмент, за допомогою якого можливо формально порівнювати результати минулих ігор для того, щоб навчити

алгоритм досягати кращих результатів. Фактично, це універсальний парамент для алгоритмів навчання з підкріпленням – багато різних функцій можуть бути використані. Для вирішення проблеми гри в морський бій, буде використано дану функцію:

$$r(a;t_0) = \sum_{t \geq t_0} \left(h(t) - \overline{h(t)} \right) (0.5)^{t-t0}.$$

Функція працює наступним чином: взявши хід a, який було зроблено в хід t0, повертає зважену суму попадань h(t) для цього і наступних кроків у грі. Таким чином, h(t) = 1 якщо постріл у хід у був успішним та дорівнює 0 в іншому випадку.

Також потрібно зазначити, що дана функція також заохочує дослідження – якщо постріл буде успішним через два кроки, нагорода досягне значення 0,5, якщо через три – 0,25, відповідно. Це заохочує до розвідки та “відкриття” недосліджених частин дошки.

Інша частина функції це віднімання $\overline{(h(t))}$. Це очікувані винагороди, які випадкова мережа отримає. Віднімаючи їх, ми винагороджуємо лише досягнення (стиль гри), які краще за повністю випадкові постріли, що призведе до пришвидшення процесу навчання.

2.3 Градієнтний спуск

Для того, щоб натренувати алгоритм грати оптимально, було додано градієнтний спуск. Параметри мережі θ можуть змінюватися в деякому кроці гри. Градієнт, являючи собою середнє усіх можливих дій, являє собою:

$$\hat{g}_i = r(a_i) \nabla_\theta \log p(a_i | s_i, \theta).$$

де p(a) – значення вірогідності обраної дії. Зазвичай неможливо обчислити, ai – дія, яку було обрано, r(ai) – винагорода, отримана за дію.

Висновки до розділу

В даному розділі було змодельовано мережу, градієнтний спуск та визначено функцію винагороди. Разом вони становлять інтелектуальну складову інтелектуальної ігрової системи та відповідають за прийняття рішень штучним інтелектом.

3 СКЛАД ТЕХНОЛОГІЧНИЙ ОПИС РІШЕННЯ

3.1 Огляд мов програмування для розробки інтелектуальної системи

На даний момент існує небагато мов програмування, які широко застосовуються для розробки в сфері машинного навчання. В даному розділі будуть оглянуті всі можливі варіанти та з них обрано найкращий

Python

Це проста в опануванні, потужна мова програмування. Розробникові, який використовує дану мову програмування, надаються можливість використовувати ООП та високо рівневі структури даних, такі як хеш-таблиця та інші. Крім того Python забезпечує підтримку ефективного функціонального програмування вбудованими бібліотеками. Швидкість розробки, простота освоєння також є перевагами.

Інтерпретатор Python і велика стандартна бібліотека вільно доступні у вихідному або довічним вигляді для всіх основних платформ на веб-сайті Python і можуть вільно поширюватися. На цьому ж сайті розміщені роздачі і посилання на багато безкоштовні сторонні модулі Python, програми та інструменти, а також додаткова документація.

Інтерпретатор Python легко розширюється за рахунок нових функцій і типів даних, реалізованих на C або C ++ (або іншими мовами, що викликаються з C). Python також підходить в якості мови розширення для налаштовуваних додатків.

Python найбільш поширений серед розробників систем машинного навчання – 57% дослідників даних та спеціалістів з машинного навчання використовують у роботі та 33% пріоритизують у розробці саме Python. Легкість у розробці, відкриті та постійно оновлювані бібліотеки для розробки, такі, наприклад, як розроблювана Google TensorFlow роблять Python пріорітетним вибором для розробки в сфері машинного навчання.

R

R – це мова програмування і програмне середовище для статистичного аналізу, графічної вистави і звітності. R був створений Россом Іхакой і Робертом Джентльменом з Університету Окленда, Нова Зеландія, і в даний час розробляється основною групою розробників R. Ядром R є комп'ютерна мова, що інтерпретується, яка дозволяє виконувати галуження і цикли, а також модульне програмування з використанням функцій.

R забезпечує ефективність інтеграції з процедурами, написаними на мовах C, C ++, .Net, Python або FORTRAN. R знаходиться у вільному доступі під GNU General Public License, і заздалегідь скомпільовані двійкові версії надаються для різних операційних систем, таких як Linux, Windows і Mac. R – це безкоштовне програмне забезпечення, поширюване під лівою копією в стилі GNU, і офіційна частина проекту GNU під назвою GNU S. Еволюція R Спочатку R був написаний Россом Іхакой і Робертом Джентльменом на факультеті статистики університету Оклендського в Окленді Нова Зеландія. R вперше з'явився в 1993 році. З середини 1997 року існує основна група («R Core Team»), яка може змінювати архів вихідної коди R.

Особливості R Як вказувалося раніше, R є мовою програмування і програмним середовищем для статистичного аналізу, графічної вистави і звітності. R – це добре розроблена, проста і ефективна мова програмування, яка включає умовні вирази, цикли, призначені для користувача рекурсивні функції і засоби введення і виводу. R володіє ефективним засобом обробки і зберігання даних, R надає набір операторів для обчислень над масивами, списками, векторами і матрицями. R надає великий, погоджений і інтегрований набір інструментів для аналізу даних. R надає графічні засоби для аналізу даних і відображення або безпосередньо на комп'ютері, або для друку на папері.

JavaScript

JavaScript – це динамічна мова програмування. Він легкий і частіше

всього є використовуваним для розробки веб-сторінок, проте може використовуватися для розробки серверної логіки або десктопних застосувань. Застосування у веб-розробці в якості засобу для розробки фронтенду дозволяє генерувати динамічні сторінки з високою якістю контенту та високою швидкістю взаємодії з терміналом користувача, що б це не було – смартфон, планшет, ноутбук чи десктоп. Будучи інтерпретуємою мовою, JavaScript є також мовою з с об’єктно-орієнтированими можливостями.

Спочату JavaScript був відомий як LiveScript, але Netscape змінив назву на JavaScript, можливо, через Java. JavaScript вперше з’явився в Netscape 2.0 в 1995 році під назвою LiveScript. Основу поклала Netscape, Internet Explorer та інші веб-браузери.

Специфікація ECMA-262 визначає стандарт мови JavaScript.

- JavaScript це легка інтерпретуєма мова програмування;
- Визначений для створення веб-застосувань;
- Доповнює та має гарну інтеграцію Java;
- Доповнює та має гарну інтеграцію з HTML;
- Відкрита та кросплатформенна.

Клієнтський JavaScript

Клієнтський JavaScript є найбільш поширеною формою мови. Сценарій має бути включений в HTML-документ або посилатися на нього для коду, який бути інтерпретованим браузером.

Це означає, що веб-сторінка не обов'язково має бути статичним HTML, але може включати програми, які взаємодіють з користувачем, управляють браузером і динамічно створюють HTML-контент.

Механізм на стороні клієнта JavaScript надає безліч переваг в порівнянні з традиційними сценаріями на стороні сервера CGI. Наприклад, ви можете використати JavaScript, щоб перевірити, чи ввів користувач діючу адресу електронної пошти в поле форми.

Код JavaScript виконується, коли користувач відправляє форму, і тільки якщо усі записи вірні, вони будуть відправлені на веб-сервер.

JavaScript може використовуватися для захоплення подій, що ініціюються користувачем, таких як натиснення кнопок, навігація по посиланнях і інші дії, які користувач ініціює явно або неявно.

Переваги JavaScript:

- менше взаємодії з сервером – ви можете перевірити введення користувача перед відправкою сторінки на сервер. Це економить серверний трафік, що означає менше навантаження на ваш сервер;
- негайний зворотний зв'язок з відвідувачами – їм не треба чекати перезавантаження сторінки, щоб дізнатися, чи забули вони щось ввести;
- підвищена інтерактивність – ви можете створювати інтерфейси, які реагують, коли користувач наводить на них курсор миші або активує їх за допомогою клавіатури.
- багатіші інтерфейси. Ви можете використати JavaScript для включення таких елементів, як компонентиперетягання і повзунки, щоб надати відвідувачам вашогосайту багатий інтерфейс.

Обмеження JavaScript:

Ми не можемо розглядати JavaScript як повноцінну мову програмування. У нім відсутні наступні важливі функції:

- Клієнтський JavaScript не дозволяє читати чи здійснювати процедуру запису у клієнтські файли. Це потрібно для виконання вимог безпеки тому, що для зловмисників було б дуже зручно мати можливість це робити при завантаженні сторінок користувачів.
- JavaScript не може використовуватися для мережевих застосувань, тому що такої підтримки немає.
- JavaScript не має багатопоточності або многопроцессорности.

Инструменты разработки JavaScript

Однією з основних переваг JavaScript є те, що він не вимагає дорогих інструментів розробки. Ви можете розпочати з простого текстового редактора, такого як Блокнот. Оскільки це мова, що інтерпретується, в контексті веб-браузеру, вам навіть не треба купувати компілятор.

Щоб зробити наше життя простіше, різні постачальники придумали дуже хороші інструменти редагування JavaScript. Деякі з них перераховані тут:

- Microsoft FrontPage – Microsoft розробила популярний редактор HTML під назвою FrontPage. FrontPage також надає веб-розробникам ряд інструментів JavaScript для допомоги в створенні інтерактивних веб-сайтів.
- Macromedia Dreamweaver MX – Macromedia Dreamweaver MX – дуже популярний редактор HTML і JavaScript серед професіоналів в області веб-розробки. Він надає декілька зручних вбудованих компонентів JavaScript, добре інтегрується з базами даних і відповідає новим стандартам, таким як XHTML і XML.
- Macromedia HomeSite 5 – HomeSite 5 – це популярний редактор HTML і JavaScript від Macromedia, який можна використати для ефективного управління особистими

Проте, попри всі гарні сторони, ця мова програмування не має достатньо засобів та бібліотек для розробки в області машинного навчання, а також немає достатньої кількості спеціалістів, які змогли б продовжити розробку в області машинного навчання на даній мові програмування.

C++

C++ це що статично типізуєма, компільова, універсальна, чутлива до регістра мова програмування вільної форми, яка підтримує процедурне, об'єктно-орієнтоване і універсальне програмування.

C++ розглядається як мова середнього рівня, оскільки вона включає

поєднання як мовних, так і низькорівневих функцій.

C++ була розроблена Бьярном Страуструпом, починаючи з 1979 року в Bell Labs в Мюррей-Хилл, штат Нью-Джерсі, в якості розширення мови C і спочатку називався C with Classes, але пізніше він був перейменований в C++ в 1983 році.

C є надмножиною C, і що практично будь-яка легальна програма на C є легальною програмою на C .

C++ повністю підтримує об'єктно-орієнтоване програмування, включаючи чотири стовпи об'єктно-орієнтованої розробки :

- Інкапсуляція
- Приховання даних
- Спадкоємство
- Поліморфізм
- Стандартні бібліотеки

Стандарт C складається з трьох важливих частин:

- Основна мова, що дає усі будівельні блоки, включаючи змінні, типи даних і літерали і т.д.;
- Стандартна бібліотека C, надає багатий набір функцій для роботи з файлами, рядками і т. д.;
- Стандартна бібліотека шаблонів (STL), що надає багатий набір методів, що управляють структурами даних і т. д.

Складність розробки та кількість коду, складність освоєння мови програмування, роблять дану мову непридатною для розробки та підтримки інтелектуальної ігрової системи.

Java

Мова програмування Java спочатку розроблялася компанією Sun Microsystems, яку ініціював Джеймс Гослінг і випустив у 1995 році як основний компонент платформи Java (Java 1.0 (J2SE)) Sun Microsystems.

Останній випуск стандартної версії Java – це Java SE 8. З розвитком Java та його широкої популярності було створено кілька конфігурацій для різних типів платформ. Наприклад: J2EE для корпоративних додатків, J2ME для мобільних додатків.

Нові версії J2 були перейменовані на Java SE, Java EE та Java ME, відповідно. Ява, як гарантовано, буде писати один раз, запустити будь-де.

Java є

Object Oriented – У Java все є об'єктом. Java можна легко розширити, оскільки вона заснована на моделі Object.

Platform Independent – на відміну від багатьох інших мов програмування, включаючи C і C ++, коли Java скомпільована, вона не компілюється в платформі, а не в платформо-байтовому коді. Цей байтовий код поширюється через Інтернет та інтерпретується віртуальною машиною (JVM) на будь-якій платформі, на якій вона запускається.

Простий – Java розроблена, щоб бути легко вивченою. Якщо ви розумієте основну концепцію OOP Java, її легко освоїти.

Безпечна – За допомогою безпечної функції Java це дає змогу розробляти системи, що не містять вірусів. Методи аутентифікації засновані на шифруванні з відкритим ключем.

Архітектура-нейтральна – Java-компілятор породжує формат об'єктно-архітектурно-нейтрального об'єкта, що робить скомпільований код виконуваним на багатьох процесорах за наявності системи виконання runtime.

Портативна – Будучи архітектурно-нейтральною і не має залежних від виконання функцій специфікації, вона є портативною. Компілятор в Java

написаний в ANSI C з чіткою межею переносу, який є підмножиною POSIX.

Надійність – Java докладає зусиль для усунення ситуацій, пов'язаних із помилками, підкреслюючи головним чином перевірку помилок часу компіляції та перевірку часу виконання.

Багатопотоковість – За допомогою багатопотокового функціоналу Java можна писати програми, які можуть виконувати багато завдань одночасно. Ця конструктивна функція дозволяє розробникам створювати інтерактивні додатки, які можуть працювати без проблем.

Інтерпретованість – байткод Java переводиться на льоту на місцеві інструкції машини і ніде не зберігається. Процес розробки є більш швидким та аналітичним, оскільки зв'язування є додатковим і легким процесом.

Висока продуктивність. Завдяки використанню компіляторів Just-In-Time, Java забезпечує високу продуктивність.

Розподілений – Java призначений для розподіленого середовища Інтернету.

Dynamic – Java вважається більш динамічною, ніж C або C ++, оскільки вона призначена для адаптації до середовища, що розвивається. Програми Java можуть нести велику кількість відомостей про виконання, які можуть бути використані для перевірки та вирішення доступу до об'єктів під час виконання.

Обмеженнями Java є застарілий синтаксис, що не надає можливості швидко розробляти застосування, та невелика популярність, як і випадку з R, що приводить до ускладнення розробки та підтримки застосувань в області машинного навчання.

Lisp

Джон МакКарті винайшов LISP в 1958 році, незабаром після розробки FORTRAN. Вперше реалізований можливостями Стіва Рассела на комп'ютері IBM 704.

Це особливо підходить для програм штучного інтелекту, оскільки він ефективно обробляє символічну інформацію.

Загальний Lisp виник у 1980-х і 1990-х роках, спробувавши уніфікувати роботу кількох груп впровадження, які були наступниками Maclisp, як ZetaLisp та NIL (New Implementation of Lisp) тощо.

Це служить загальною мовою, яку можна легко розширити для конкретної реалізації.

Програми, написані на Common LISP, не залежать від специфічних для машини характеристик, таких як довжина слова та ін.

Вона використовує ітеративну методологію проектування та легку розширюваність. Гарні сторони Lisp:

- Це дозволяє динамічно оновлювати програми;
- Це забезпечує налагодження на високому рівні;
- Це забезпечує покращений об'єктно-орієнтований програмування;
- Це забезпечує зручну макросистему;
- Він надає широкомасштабні типи даних, наприклад, об'єкти, структури, списки, вектори, регульовані масиви, хеш-таблиці та символи;
- Це забезпечує підтримку ООП парадигми;
- Він забезпечує повну бібліотеку вводу-виводу;
- Він забезпечує широкі можливості щодо створення структур.

Обмеження Lisp: хоча легкість обробки символьної інформації робить мову легкою для розробки деяких видів штучного інтелекту, у випадку

обробки фактично масиву, який собою представляє ігрова дошка у грі морський бій не забезпечає перевагу саме тут. По-друге, як вже зазначалося, будучи одним із найнепопулярніших мов програмування, Lisp забезпечає труднощі у пошуці розробників. По-третє, відсутність розробників забезпечує повільність розробки бібліотек та їх підтримку для розробки застосувань у сфері штучного інтелекту. Порівнюючи Lisp з Python, останній виграє у простоті написання коду, легкості освоєння, популярності та потенціалові для розвитку у майбутньому, та підтримці корпорацій, що продовжують писати застосування, а, отже, і розвивати мову програмування Python зараз та в майбутньому.

Висновок щодо вибору мови програмування

Обираючи за критеріями популярності, що призводить до більшої кількості розробників, які, потенційно, можуть бути залученими до подальшої розробки, та можливості подальшого розвитку мови програмування, що призведе до появи кращих можливостей для розробки в майбутньому, було обрано мову програмування Python.

Другим критерієм підтримка та можливості подальшого розвитку, що засвідчує факт того, що провідна бібліотека програмного коду для розробки в області машинного навчання, TensorFlow, розроблюється та підтримується Google. Факт підтримки, що веде до наслідків у вигляді гарантованого розвитку та покращенню існуючих засобів (наприклад, виправлення помилок, оптимізація).

3.2 Формалізація інтелектуальної системи у вигляді uml діаграм прецедентів

Загальний опис

UML – спосіб уніфіковано описувати ПЗ так, щоб основні процеси були відображені зрозуміло та окремо від їх динамічної поведінки, що також можливо описати за допомогою UML. Формалізація описання ПЗ необхідна для того, щоб розробники ефективніше створювали системи програмного забезпечення за допомогою кращого розуміння технічних вимог, роботи вже створених іншими розробниками систем та швидше навчалися розуміти складні системи під час безпосереднього навчання роботи з ними та розробки цих складних в плані архітектури ПЗ систем.

Крім того, UML можливо використовувати не тільки для експлуатації розробниками, а й для розуміння бізнес-процесів не причетними до розробки ПЗ людьми, а також тими, хто з розробниками взаємодіє, наприклад, прожект-менеджерам, що контролюють стан виконання розробниками плану по розробці ПЗ.

Частіше всього UML використовують для описання ПЗ, створеного за парадигмою об’єктно-орієнтованого програмування.

Розроблення діаграм прецедентів

Для створення інтелектуальної ігрової системи в першу чергу визначаються вимоги, яким повинна задовольняти система.

Для формалізації ПЗ, все частіше використовується опис функціональності системи через ДВВ. Варіанти використання це опис послідовності дій, що може здійснювати система у відповідь на зовнішні впливи користувачів або інших програмних систем. Варіанти використання відображають функціональність системи з точки зору отримання результату для користувача, тому вони точніше дозволяють поділяти функції за значимістю одержуваного результату [1, 22].

За допомогою діаграм прецедентів частіше всього приходить визначення вимог (в більшості випадків функціональних) до ПЗ.. Під час аналізу і проектування варіанти використання дозволяють зрозуміти як результати, які хоче отримати користувач впливають на архітектуру системи і як повинні поводитися компоненти системи, для того щоб реалізувати потрібну для користувача функціональність.

ДВВ складається з акторів, для яких система дає потенційні можливості, крім того ДВВ містить UC, що описують те, що актор хоче отримати від системи [14,15].

Між акторами і варіантами використання можуть бути різні види взаємодії.

Створена діаграма прецедентів описує систему “Ігрова система” і в неї включені всі прецеденти, які присутні в даній системі, включаючи розширення основних прецедентів, таких як “Повідомлення про помилку” та “Повідомлення про неправильний ввід даних”, що довзоляє зрозуміти логіку функціональної частини даної системи.

Рисунок 3.1 – Діаграма прецедентів системи «Ігрова система»

У наступних таблицях буде розглянуто докладний опис усіх ВВ, що наявні на розробленій ДВВ.

Прецедент «Виконання команд», додатковий опис якого зображено у табл. 4.1, являє собою дії по виконанню команд.

Таблиця 3.1 – Опис прецеденту «Виконання команд»

Назва (UC)	Виконання команд
Мета або контекст використання	Виконання команд керування, що були здійснені з клавіатури гравцем
Передумова	Отримано команди.
Умова успішного виконання	Коректна команда.

Умова невиконання	Помилкова команда.	
Первинний, вторинний актор	Ігрова система	
Тригер	Відсутній	
	Кроки	*Дія*
	1	Отримання команд.
	2	Виконання команд.
Розширення (extention)	Повідомлення про помилку.	

ВВ «Повідомлення про помилку» табл. 4.2, являє собою розширення для прецедентів «Виконання команд керування», «Збереження нейронної мережі на жорсткий диск», «Завантаження нейронної мережі з жорсткого диску» яке надає інформацію про помилку користувачеві.

Таблиця 3.2 – Опис прецеденту «Повідомлення про помилку»

Назва (UC)	Повідомлення про помилку	
Мета або контекст використання	Відправлення інформації про стан справності	
Передумова	Виконання формування інформації про стан.	
Умова успішного виконання	Достатній рівень доступу користувача.	
Умова невиконання	Недостатній рівень доступу користувача.	
Актори	Ігрова система.	
Тригер	Неможливість виконання будь-якогї зі з’єднаних на діаграмі дій	
Опис(description)	*Кроки*	*Дія*
	1	Отримання сформованого стану.
	2	Надання користувачеві інформації про стан.

Прецедент «Обробка вводу користувача», додатковий опис якого зображено у табл. 3.3, являє собою дії по обробці отриманих команд.

Таблиця 3.3 – Опис прецеденту «Обробка команд керування»

Назва (UC)	Обробка команд керування	
Мета або контекст використання	Обробка команд керування, що були зроблені користувачем	
Передумова	Отримано команди керування.	
Умова успішного виконання	Коректна команда управління.	
Умова невиконання	Помилкова команда.	
Первинний, вторинний актор	Ігрова система.	
Тригер	Відсутні	
	Кроки	*Дія*
	1	Отримання команд керування.
	2	Початок виконання команди.
Розширення (extention)	Повідомлення про неправильний ввід.	

ВВ «Завантаження нейронної мережі з жорсткого диску» табл. 3.4, додатковий опис якого зображено у табл. 3.3, являє собою дії по завантаженню нейронної мережі з жорсткого диску.

Таблиця 3.4 – Опис прецеденту «Завантаження нейронної мережі з жорсткого диску»

Назва (UC)	Завантаження нейронної мережі з жорсткого диску	
Мета або контекст використання	Завантаження нейронної мережі з жорсткого диску для подальшого використання нейронної мержеі	
Передумова	Отримано команди керування.	
Умова успішного виконання	Коректна команда управління.	
Умова невиконання	Помилка завантаження.	
Первинний, вторинний актор	Ігрова система.	
Тригер	Відсутні	
	Кроки	*Дія*
	1	Отримання команд керування.
	2	Завантаження нейронної мережі.
Розширення (extention)	Повідомлення про помилку.	

ВВ «Збереження нейронної мережі на жорсткий диск» табл. 3.5, додатковий опис якого зображено у табл. 3.5, являє собою дії по завантаженню нейронної мережі з жорсткого диску.

Таблиця 3.5 – Опис прецеденту «Збереження нейронної мережі на жорсткий диск»

Назва (UC)	Збереження нейронної мережі на жорсткий диск	
Мета або контекст використання	Збереження нейронної мережі на жорсткий диск для подальшого використання нейронної мержеі	
Передумова	Отримано команди керування.	
Умова успішного виконання	Коректна команда управління.	
Умова невиконання	Помилка збереження.	
Первинний, вторинний актор	Ігрова система.	
Тригер	Відсутні	
	Кроки	*Дія*
	1	Отримання команд керування.
	2	Збереження нейронної мережі.
Розширення (extention)	Повідомлення про помилку.	

ВВ «Повідомлення про неправильний ввід» табл. 3.6, являє собою розширення для прецедентів «Виконання команд керування», «Збереження нейронної мережі на жорсткий диск», «Завантаження нейронної мережі з жорсткого диску» яке надає інформацію про помилку користувачеві.

Таблиця 3.6 – Опис прецеденту «Повідомлення про помилку»

Назва (UC)	Повідомлення про неправильний ввід	
Мета або контекст використання	Надання користувачеві інформації про неправильний ввід	
Передумова	Отримання вводу.	
Умова успішного виконання	Достатній рівень доступу користувача.	
Умова невиконання	Недостатній рівень доступу користувача.	
Актори	Ігрова система.	
Тригер	Ввід користувача неправильний або неможливо інтерпретувати	
Опис(description)	*Кроки*	*Дія*
	1	Отримання команди.
	2	Надання користувачеві інформації про неправильний ввід.

3.3 Формалізація поведінки інтелектуальної системи за допомогою UML діаграм

UML передбачає використання декількох видів діаграм, що забезпечують всебічний опис характеристик, поведінки та можливостей взаємодії кінцевих користувачів, частин системи, а також персоналу з розширеними можливостями, наприклад, системному адміністраторові бази даних інформаційної системи. Це передбачає можливість відслідковування на відповідній діаграмі зміни стану об’єкту, навіть якщо вона викликана зовнішніми факторами.

Діграма застосувань використовується для того, щоб описати можливі стани системи у контексті застосування системою реальним користувачем, тобто показати систему з ракурсу застосування.

Діаграми станів знакходять використання у процесі опису поведінки частин системи, проте можливо їхнє використання і для опису процесу функціонування інших частин системи, наприклад, можливо використання для процесу опису акторів системи, підсистем системи, операцій системи, так методів, які є у обраному ПЗ.

ДКА в мові UML є деякою абстракцією, призначеною для моделювання системи та її складових, в тому числі і їх поведінки.

UML діаграма застосувань

Діаграма застосувань призначена для формалізації можливостей застосування системи групами користувачів. В даному випадку користувач тільки один. На рисунку 3.2 зображена діаграма застосувань. Опис застосувань наведено у таблицях 3.7, 3.8, 3.9.

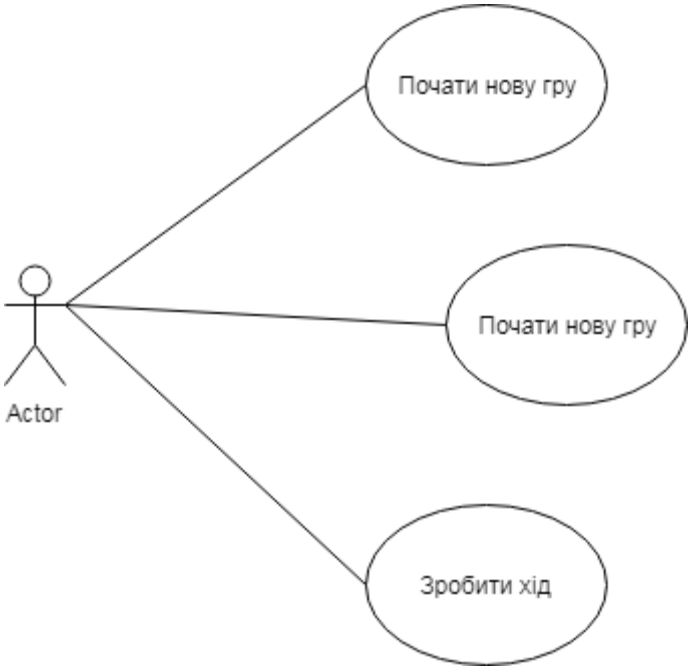

Рисунок 3.2 – Діаграма застосувань

Таблиця 3.7 – Опис застосування «Почати нову гру»

Назва (UC)	Почати нову гру
Мета або контекст використання	Надання користувачеві можливості почати грати
Передумова	Отримання вводу, функціонування системи
Умова успішного виконання	-
Умова невиконання	-.
Актори	Гравець

Тригер	Команда користувача	
Розширення (extention)	1	Здатися
Розширення (extention)	2	Зробити хід

Вид застосування “почати нову гру” – основна можливість для гравця. Після відповідної команди система починає нову партію.

Таблиця 3.8 – Опис застосування “Здатися”

Назва (UC)	Почати нову гру
Мета або контекст використання	Надання користувачеві можливості здатися
Передумова	Отримання вводу, функціонування системи
Умова успішного виконання	Правильна команда
Умова невиконання	-
Актори	Гравець
Тригер	Команда користувача

Вид застосування “Здатися” – можливість для гравця припини партію. Після отримання інтелектуальною ігровою системою відповідної команди гравець переноситься у головне меню и має можливість почати нову гру.

Таблиця 3.9 – Опис застосування “Зробити хід”

Назва (UC)	Зробити хід
Мета або контекст використання	Надання користувачеві можливості зробити хід
Передумова	Отримання вводу, функціонування системи
Умова успішного виконання	Валідні координати пострілу
Умова невиконання	Неправильно введені координати пострілу
Актори	Гравець
Тригер	Команда користувача

Застосування “Зробити хід” – одне із основних в інтелектуальній ігровій системі. Умовою успішного виконання застосування є координати, які можливо обробити системою, в той час як неправильний ввід призводить до помилки та повідомлення від системи про необхідність ввести правильні координати.

UML діаграма послідовностей

Розроблення загальної діаграми послідовності

Загальна діаграма послідовності містить 3 основні об’єкти, між якими відбувається взаємодія, а саме:

- користувач;
- ігрова система;
- нейронна мережа.

Користувач являє собою будь яку людину, що вмикає ігрову систему, Ігрова система – це програма, яку використовує користувач для гри. Нейронна мережі це система, яка вирішує, куда робити наступний постріл у грі Морський бій.

Опишемо дану діаграму докладніше. Після запуску ігрової системи користувач має можливість почати нову гру. Після початку гри ігрова система завантажує збережений стан нейронної мережі. Під час гри гравець та нейронна мережа почергово дають ігровій системі координати місць пострілів, що відображається у гравця у інтерфейсі у вигляді оновлення ігрової дошки. Гравець може завчасно закінчити гру, або закінчити її, програвши або здобувши перемогу. Після цього стан нейронної мережі зберігається до наступного використання

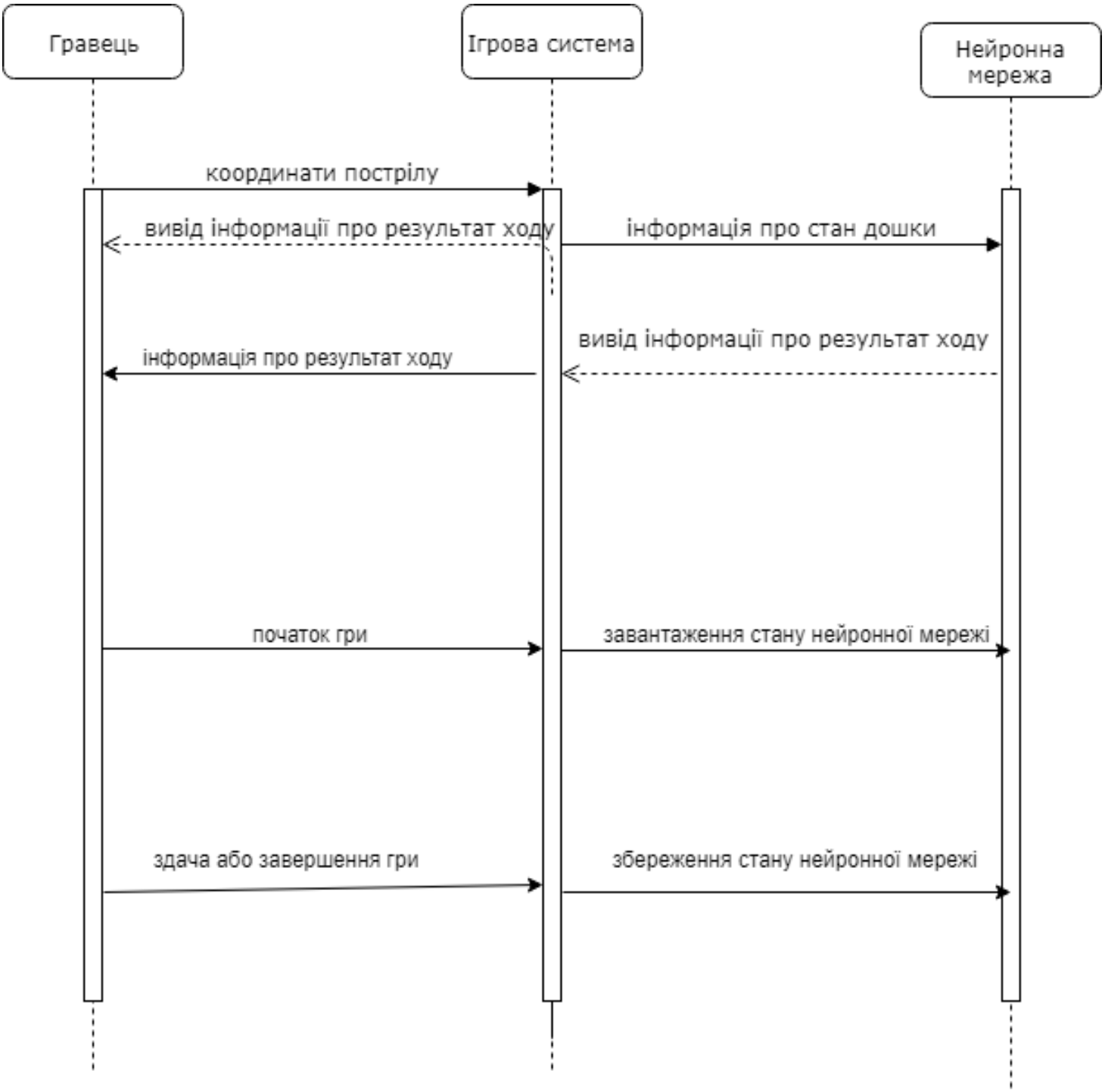

Рисунок 3.3 – Загальна діаграма послідовності

Розробка UML діаграм діяльності

Діагрма діяльності призначена для описання динамічних аспектів системи. В загальному випадку, це можливість представити перехід від одної дії до іншої. Активність можна визначити як застосування системи[18,19].

Основна ціль діаграми діяльності це отримати динаміку поведінки системи. Активність і є частиною функціонування системи. Єдиним обмеженням діаграми активності є те, що вона не відображає повідомлень, які створюються і надходять від однієї частини функціонуючої системи до іншої.

Отже, основний функціонал діграм діяльності можна визначити наступним чином:

- показати плин активностей у системі;
- описати послідовність переходу від однієї активності до іншої;
- описати паралельний, розділяючийся або паралельний перебіг подій у системі.

Діаграма діяльності “Хід гравця” (Рисунок 3.4) демонструє активний хід гравця (виключаючи процес здачі тому, що здатися можна і під час ходу ігрової системи). Гравець вводить координати місця пострілу після того, як хід переходить до нього. Після цього координати перевіряються, і, якщо координати вірні, стан ігрового поля змінюється.

Діаграма діяльності “Запис стану нейронної мережі” (Рисунок 3.5) демонструє механізм збереження стану нейронної мережі на жорсткому дискові задля можливості використання даної нейронної мережі після вимкнення робочої машини. В разі помилки гравець бачить інформацію про помилку.

Діаграма діяльності “Початок нової гри” (Рисунок 3.6) демонструє механізм початку нової партії. Стан нейронної мережі завантажується та виводиться інформація про результат цього процесу.

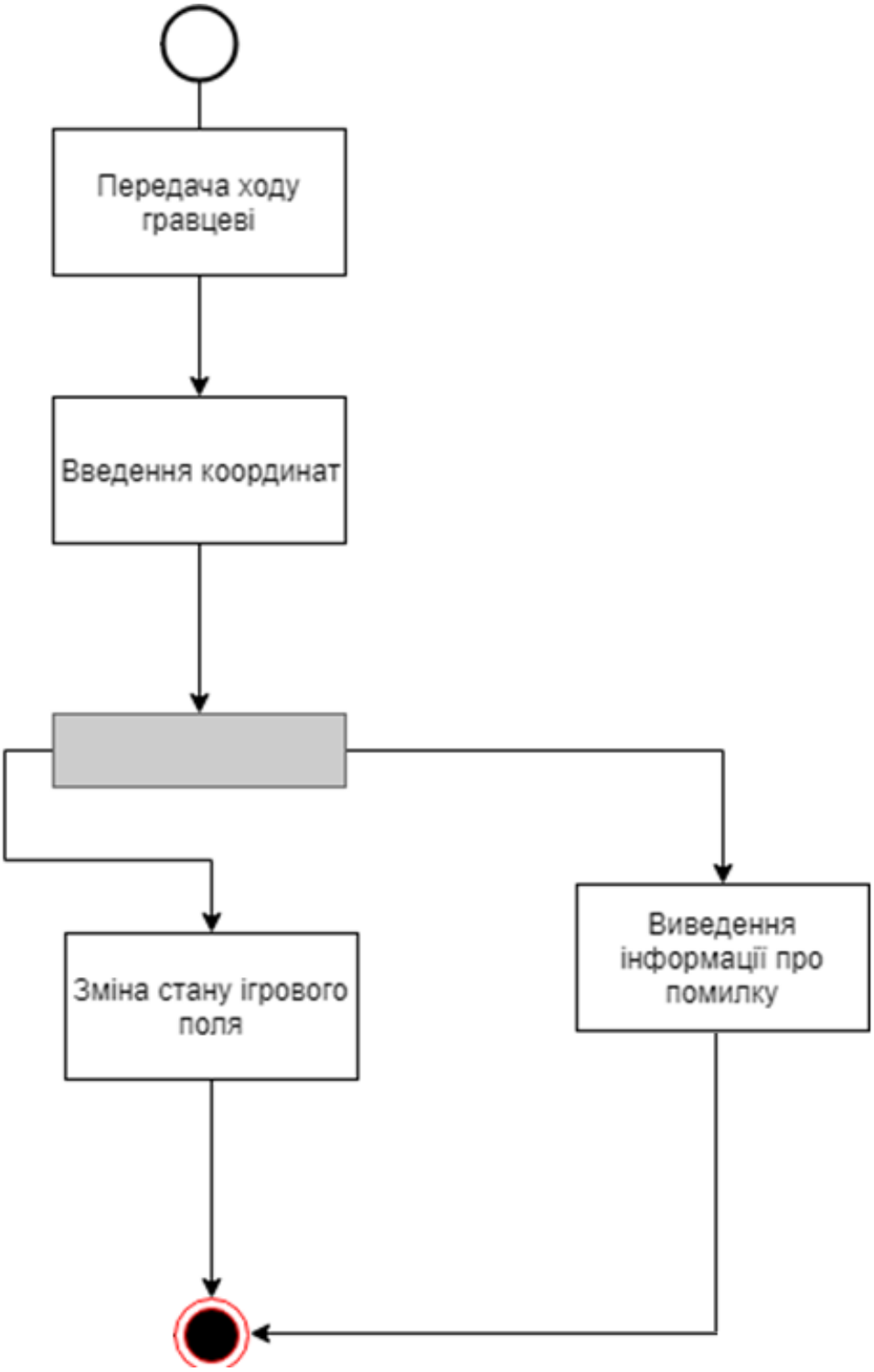

Рисунок 3.4 – Діаграма діяльності “Хід гравця”

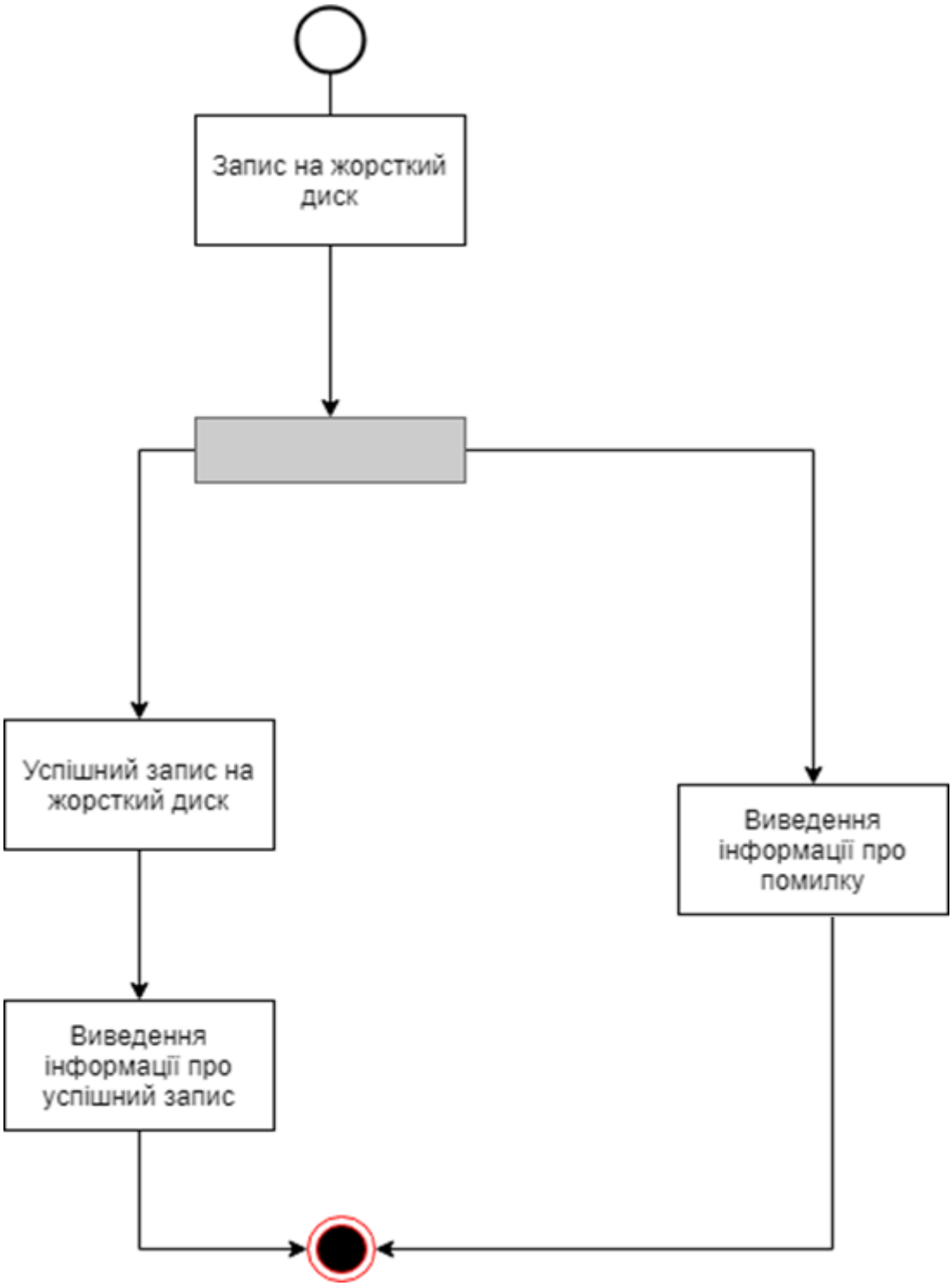

Рисунок 3.5 – Діаграма діяльності “Збереження стану нейронної мережі”

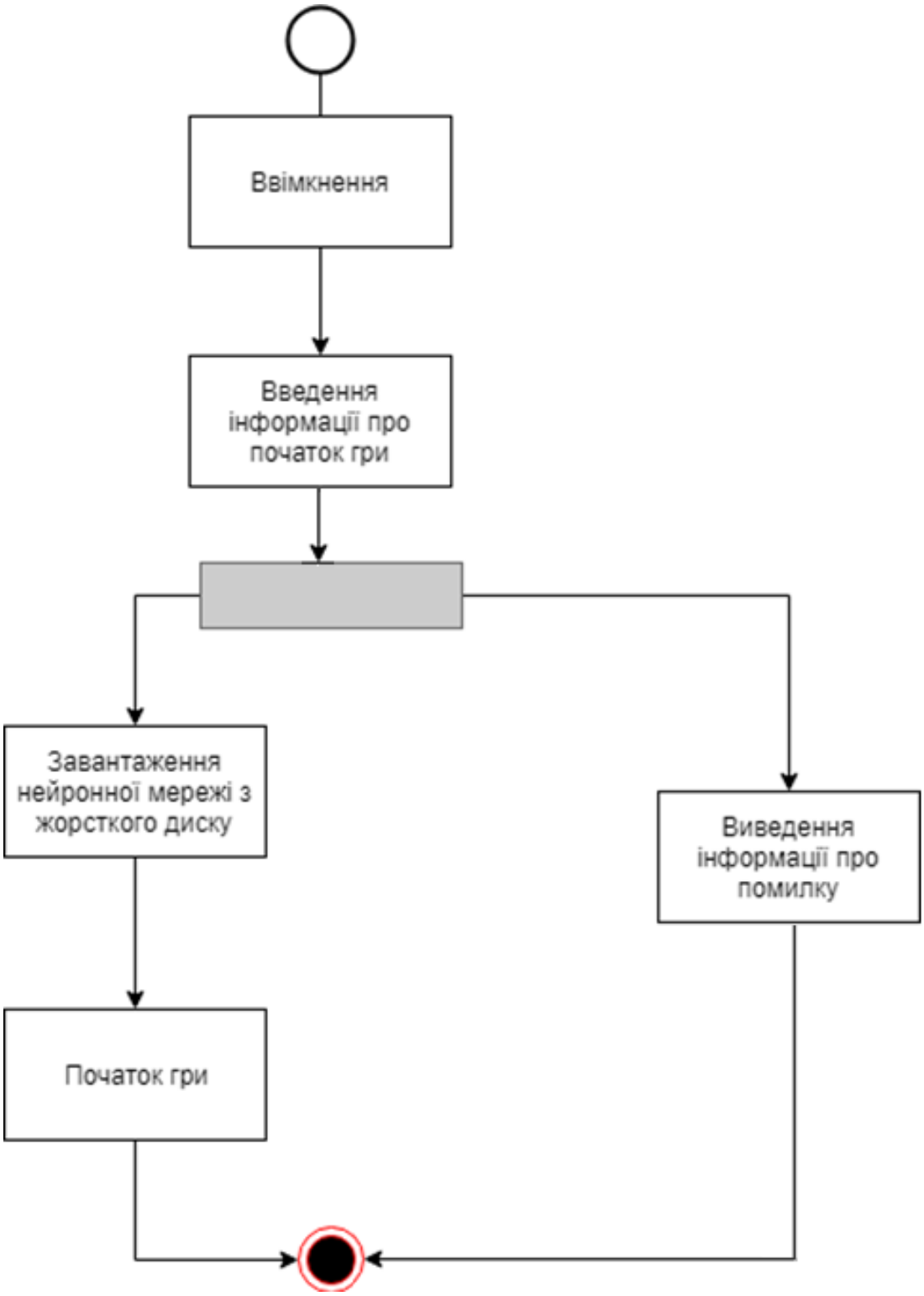

Рисунок 3.6 – Діаграма діяльності “Початок нової гри”

Діаграма діяльності “Початок нової гри” демонструє механізм початку нової партії. Стан нейронної мережі завантажується та виводиться інформація про успішний або не успішний результат цього процесу.

3.3 Розробка діаграми компонентів

Діаграма компонентів (вона ж структурна схема) дозволяє визначити склад компонентів ПЗ.

При розробці діаграм компонентів переслідуються цілі:

- специфікація загальної структури вихідного коду системи;
- специфікація виконуваного варіанту системи.

Дана діаграма забезпечує узгоджений перехід від логічного до фізичного поданням системи у вигляді програмних компонентів. Основними елементами діаграми є компоненти, інтерфейси і залежності між ними. Крім цього, на ній можуть відображатися ключові класи, що входять в компоненти.

Компонент – це фізична частина системи. Компоненти, що представляють собою файли з вихідним кодом класів, бібліотеки, виконувані модулі і т.п., які повинні володіти узгодженим набором інтерфейсів. Для їх графічного представлення використовуються наступні графічні символи

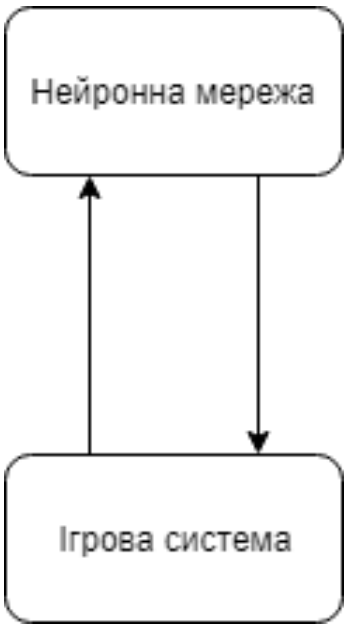

Рисунок 3.7 – Діаграма компонентів

Діаграма компонентів ігрової інтелектуальної системи демонструє, що вона складається з двох компонентів – ігрової системи та нейронної мережі[21].

Ігрова система складається с системи вводу-виводу (інтерпретації команд користувача), системи інформування користувача про помилки та неправильно введені координати або команди, системи зв’язку з нейронною мережею.

Нейронна мережа, в свою чергу, складається з трьох скадових – самої мережі, градієнтного спуску та функції винагороди. Нейронна мережа скадається з трьох шарів (вхідний, прихований, вихідний).

Висновки до розділу

В даному розділі було обрано мову програмування Python та були розроблені діаграма прецедентів, спрощена модель нейронної мережі, загальна діаграма послідовності, діаграма застосувань, діаграма компонентів, діаграми діяльності.

4 ТЕСТУВАННЯ РІШЕННЯ

Для того, щоб показати ефективність методу машинного навчання, а точніше навчання з підкріпленням, було проведено заміри ефективності інтелектуальної ігрової системи під час навчання. Графік, показаний нижче, демонструє швидкість навчання системи для виконання завдання по бомбуванню одного корабля на меленькому полі. Для поравняння обрано такі параметри: довжина гри (час, необхідний для повного знищення корабля) відносно кількості ітерацій навчання.

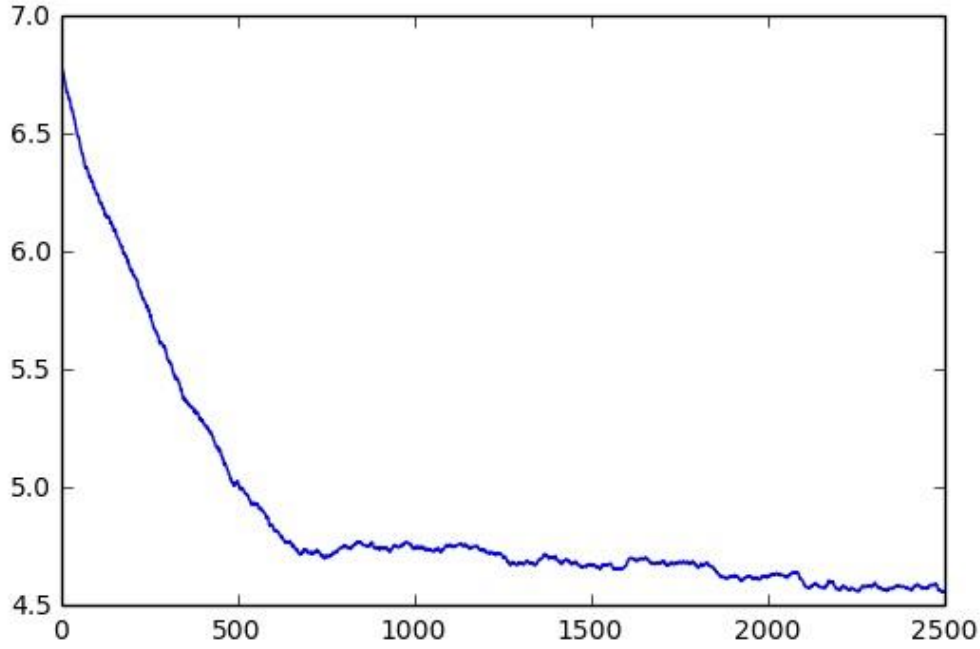

Рисунок 4.1 – Графік навчання системи

Наведений вище графік показує, що на початку система швидко опановує основи, а потім продовжує покращувати свої результати.

Суть автоматизованого тестування полягає в написанні коду, який буде перевіряти функціональність системи на відповідність вимогам. Тести запускаються кожен раз перед запуском або публікацією системи. Середовище розробки Pycharm дозволяю у зручному вигляді контролювати процес виконання тестів

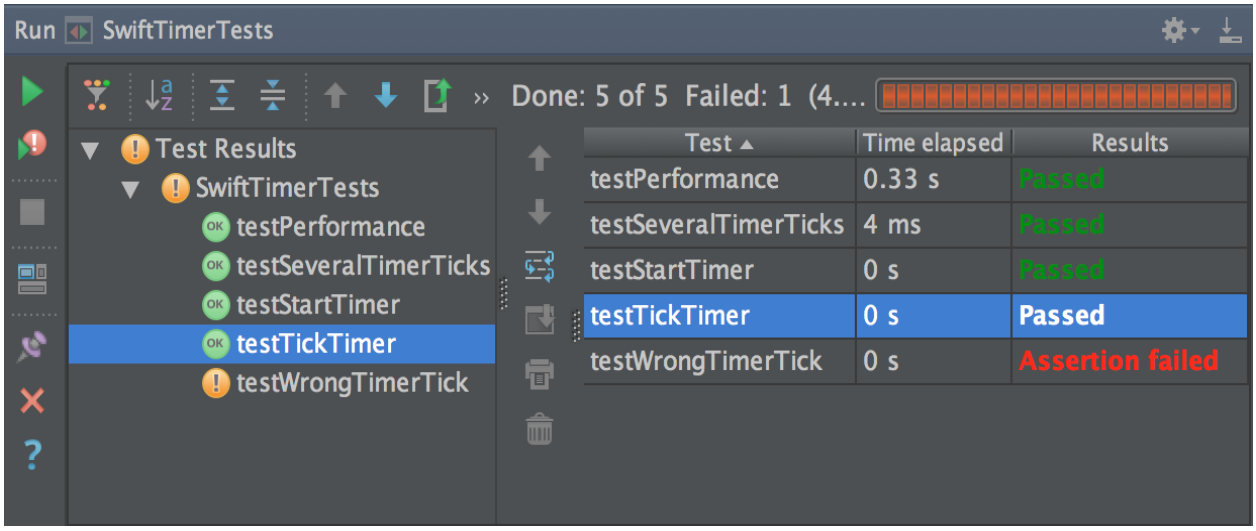

Рисунок 4.1 – Вікно відображення тестів в середовищі Pycharm

В даній системі автоматизоване тестування застосовувалося для тестування ігрової системи. Для перевірки стійкості служб до навантаження було проведено навантажувальне тестування на перевірку кількості можливих одночасних запитів на служби. За допомогою даного виду тестування було усунуто проблему падіння застосування при недостатній кількості оперативної пам’яті для нього.

Ручне тестування

Ручне тестування проводиться самим розробником для знаходження тих помилок, які не в змозі виявити автоматизоване тестування. Розробник імітує користувача системи і намагається знайти всі вразливі місця системи. Особливістю даного типу тестування є відсутність використання автоматичних засобів тестування і використання лише тих можливостей

взаємодії, які доступні лише звичайним користувачам.

Одним із типів тестування є тестування готовності, що перевіряє відповідність системи вимогам специфікації. Це дозволяє

Переваги логування у тому, що воно дозволяє зберегти інформацію про помилки для майбутніх розробників, а також для перевірки виправлення помилок, а також як частина

Також тестування проводилося за окремими сценаріями. Дані сценарії біло прописано завчасно. Після проведення тестування по сценаріям було зроблено висновки та проаналізовано.

Висновки до розділу

В даному розділі було показано тестування та ефективність системи. Було проведено тестування модулів системи. Тестування відбувалося в автоматизованому режимі (за допомогою юніт тестів та інтеграційних тестів) та в ручному режимі. Для визначення несправностей системи застосовується логування. Система повністю протестована та готова до експлуатації.

5 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1 Опис ідеї проекту

Метою розділу є розроблення інтелектуальної ігрової системи з використанням нейронної мережі. Основними складовими ідеї стартапу є:

- вигоди для користувача;
- напрямки застосування для ідеї;
- зміст.

Таблиця 5.1 Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Інтелектуальна ігрова система з використанням нейронної мережі.	Ігровий процес	Емоції від процесу гри.

Дана архітектура відрізняється від аналогічних більшою кількістю задоволення, яке отримує гравець, відностою дешевизною, легкістю масштабування.

Основними конкурентами на ринку є Warships, “Морський бій 2012”, WWar. Докладний огляд сильних, слабких та нейтральних характеристик проекту у порівнянні з конкурентами розглянуто на табл. 5.2

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*			
		Мій проект	Warships	Морський бій 2012	WWar
.	Економічні	50 грн	345 грн	100 грн	120 грн
.	Технологіч-ність	Відсутність графічного інтерфейсу	Мала кількість функцій	Велика функціональність	Велика функціональність
.	Надійність	Високий рівень надійності	Високий рівень надійності	Високий рівень надійності	Високий рівень надійності

Ціна, менша за ту, що є наявною у конкурентів є сильною стороною ідеї продукту, що забезпечить її перевагу над ідеями конкурентів, які не можуть конкурувати з даною сильною сторону.

В той час нейтральною стороною продукту є її надійність тому, що дана характеристика притаманна також продуктам конкурентів.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу буде проведеною переверку технологічних рішень, з використанням яких буде можливе виконання реалізації ідеї.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

n/n	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
	Створення ігрової системи	Використання технологій побудови web додатків	Python	Доступна
	Програмування нейронної мережі	Технологія для збереження авторизованого користувача	Python, TensorFlow, numpy	Доступна
Обрана технологія реалізації ідеї проекту: Проект можливо реалізувати. Ігрова система буде реалізована за допомогою Python, нейронна мережа за допомогою TensorFlow, Python, numpy				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Спочатку проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.4).

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту [24]

n/n	*Показники стану ринку (найменування)*	*Характеристика*
	Кількість основних конкурентів	2-4
	Загальний обсяг продаж, грн/ум.од	150-200 за 1 систему, за рік 1000-1500 шт
	Динаміка	Позитивна
	Обмеження входу на ринок	Без обмежень
	Специфікації сертифікації та стандартизації	Немає
	Рентабельність галузі	Дуже висока, під п’ятидесяти до восьмидесяти відсотів за одну одиницю товару.

Проаналізувавши попередню таблицю приходимо до висновку, що на даний момент часу с такими характеристиками ринку від є привабливим для входу, що означає що потрібно розглядати можливість реалізації власної системи для подальшого розповсюдження та отримання вигоди.

Таблиця 5.5 – Фактори загроз

n/n	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
	Знецінення грошової одиниці	Підвищення ціни внаслідок зменшення попиту	Знижки і акції
	Вхід нових гравців на ринок, тобто збільшення конкуренції	Зменшення долі на ринку	Реклама
	Ріст собівартості розробки	Збільшення платні працівникам, що займаються розробкою	Зміна вартості продукції для кінцевих споживачів

Таблиця 5.6 – Фактори можливостей

№ n/n	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Підвищення популярності ігрових систем	Збільшення попиту	Розробка нового функціоналу системи
2	Зниження ціни розробки	Зменшення витрат	Знаходження нових ринків збуту

Буде проведено аналіз пропозиції: буде визначено основні загальні риси конкуренції на ринку та даний момент табл. 5.8.

Таблиця 5.7 Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Олігополія	Невелика кількість виробників, переважно нецінова конкуренція	Впровадження реклами, гарантування якості продукції, реалізація сервісного обслуговування
2. Національний рівень конкурентної боротьби	Конкуренція в країні	Вихід на світовий ринок
3. Боротьма між різними галузями	Міжгалузева конкуренція	Реалізація сайту для дистанційного управління, створення додаткових індивідуальних функцій
4.Товарно-видова конкуренція	Задовольняючі потреби споживача види товарів	Введення нових функцій
5.За характером конкурентних переваг - цінова	Використанням цін як засобу досягнення кращих економічних умов збуту і ринку	Цінової політики, яка відповідає поставленим умовам конкеренції
6. За інтенсивністю - не марочна	Відсутня прив’язаність до певної марки, можливість співпраці.	Взаємодія з усіма марками

Після аналізу конкуренції проведемо більш детальний аналіз умов

конкуренції в галузі на таблиці 5.9.

Таблиця 5.8 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	Warships, Морський бій 2012, WWar	Відсутні	Відсутні	Контроль якості Чутливість до зміни цін	Ціна Змінні витати
Висновки:	Інтенсивність конкурентності низька – ринок можі вмістити більше гравців	можливість входу на ринок є; потенційні конкуренти відсутні; можливий вихід на ринок на протязі до 1 року;	Постачальники відсутні, впливу на умови роботи на ринку немає.	Умови створюються клієнтами. Вимоги, в основному висовуються до якості та ціни	Цінова політика на товари замінники складають основну частину обмежень.

Відсутність потенційних умов створює сприятливі умови роботи на ринку. Потреба клієнтів у товарах відображена у позитивній динаміці ринку.

На основі аналізу конкуренції, проведеного в таблиця 5.9, а також із урахуванням характеристик ідеї проекту (табл. 5.2), вимог споживачів до

товару (таблиці 5.6) та факторів маркетингового середовища [25] (таблиці 5.7-5.8) визначимо та обґрунтуємо перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 5.10

Таблиця 5.9 – Обґрунтування факторів конкурентоспроможності

п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
	Ціна	Споживачі є чутливими до цін.
	Якість	Важливим є реалістичність ігрового процесу.
	Довговічність	Немає потреби виконувати заміну системи.
	Масштабованість	Можливість виконувати процес замовлення різновидів системи.

Визначивши фактори конкурентоспроможності (табл. 5.10) буде провведено аналіз сильних та слабких сторін даного стартап-проекту (табл. 5.11).

Таблиця 5.10 – Порівняльний аналіз сильних та слабких сторін «назва проекту»

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Ціна	20	+						
2	Якість	11					+		
3	Довговічність	10				+			
4	Масштабованість	17		+					

Таблиця 5.11 – SWOT- аналіз стартап-проекту

Сильні сторони: Нижча за наявну у конкурентів ціна.	Слабкі сторони: Менш надійний ніж деякі конкуренти, відсутність графічного інтерфейсу
Можливості: Збільшення попиту на товар, здешевлення розробки	Загрози: Ризик появи конкурентів, зміна динаміки ринку

Друга та четверта альтернативи будуть виступати в якості обраних. Розроблення додаткових функцій є досить перспективною, та строки реалізації достатньо короткі.

Таблиця 5.12 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Здешевлення ціни на розробку	Низька	До двох місяців
2	Розроблення додаткових функцій	Висока	До трьох неділь
3	Підвищення продуктивності	Висока	До двох років
4	Запуск реклами	Середня	До десяти днів

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 6.14).

Таблиця 5.13 – Вибір цільових груп потенційних споживачів

п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
	Казуальні гравці	так	10%	Висока	Складна
	Хардкорні гравці	ні	3%	Висока	Середня
	Middlecore гравці	так	5%	Висока	Складна
У якості цільових груп обрано: казуальні та middlecore гравці					

У якості цільових груп за результатами аналізу обрано казуальних та middlecore гравців.

Таблиця 5.14 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
Стратегія виклику лідера	Стратегія диференційного маркетингу	Індивідуальний підхід до клієнта; кращий набір функцій і ціна ніж у конкурента	Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмінних властивостей, які роблять товар відмін-ним від товарів можливих конкурентів.

Таблиця 5.15 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
Ні	Буде відбуватись пошук нових та збирання існуючих споживачів	Стандартний набір буде скопійовано, але додано багато нововведень	Стратегія виклику лідера

Таблиця 5.16 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
Низька ціна, надійність, функціональність	Стратегія диференціювання	Наявність можливості самонавчання	Постійне оновлення(вдосконалення) Найкраща якість Розширюваність

5.5 Розроблення маркетингової програми стартап-проекту

Таблиця 5.17 – Визначення ключових переваг концепції потенційного

товару

п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
	Цікавість ігрового процесу	Наявність покращення результатів опонента з часом	Наявність можливості навчання
	Масштабованість системи	Просте додавання нових елементів системи	Забезпечена можливість додавання функцій
	Низька ціна	Конкурентна ціна	Ціна нижча ніж у більшості конкурентів

Таблиця 5.18 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>І.Товар за задумом</td><td colspan="3">Система для дистанційного управління рухомим об’єктом</td></tr>
<tr><td rowspan="9"></td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1.Інтерфейс для управління на сайті</td><td>М</td><td>Тх/Тл/Е</td></tr>
<tr><td>2.Універсальність для різних РО</td><td>Нм</td><td>Вр/Тх/Тл</td></tr>
<tr><td>3.Частота надсилання інформації</td><td>М</td><td>Тх/Тл</td></tr>
<tr><td>4. Якість переданого відео</td><td>М</td><td>Вр/Тх/Тл</td></tr>
<tr><td>5. Необмежений термін дії</td><td>М</td><td>Вр/Тл</td></tr>
<tr><td colspan="3">Якість: відповідає нормам розробки програмного забезпечення.</td></tr>
<tr><td colspan="3">Пакування: Встановлення системи спеціалістами фірми виробника</td></tr>
<tr><td colspan="3">Марка: ПП «Sea war 2018»</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: Товар буде запатентований</td></tr>
</table>

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар.

Таблиця 5.19 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
Відсутні	200 грн	2000 грн	50 грн – 70 грн

Таблиця 5.20 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
Купування у роздріб	Немає	Виробник – споживач	Web-сайт

Концепція маркетингових комунікацій яскраво відображає специфіку поведінки цільових клієнтів.

Таблиця 5.21 – Концепція маркетингових комунікацій1

Специфіка поведінки цільових клієнтів	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
Ретельний вибір товару, порівняння з аналогами	Відгуки в інтернет-магазині	Підтримка, індивідуальний підхід, можливість розширення	Донесення переваг до клієнтів	Новий ігровий досвід

Висновки до розділу

Ситсема має декілька сильних сторін у порівнянні з конкурентами, що забезпечить її стабільний вихід на ринок. Технології для розробки безкоштовні та легкодоступні.

Вивід продукції на ринок доцільний, ризики враховані та визнані не маючими великого значення.

Було розроблено маркетингову програму та ринкову стратегію.

Тому можна зробити висновок, що інтелектуальна ігрова система з використанням нейронної мережі є перспективною для розробки.

6 ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ІГРОВОЇ СИСТЕМИ

Комплекс технічних засобів ІП забезпечує:

- введення інформації;
- обробку і накопичення інформації;
- контроль інформації на всіх етапах її обробки;
- якісну підготовку та видачу текстової інформації;
- можливість агрегатного збільшення потужності комплексу.

Технічне забезпечення ІП буде базуватися на таких групах обладнання:

- засоби обробки та збереження даних;
- мережеві засоби передачі даних;
- сервери;
- робочі станції;

Робочі місця користувачів побудовані на основі сучасних персональних комп’ютерів із процесором Intel, 8 Гбайт оперативної пам’яті і вище, 100 ГБ на жорсткому магнітному диску і вище, монітор із розміром 18” та більше.

Висновки до розділу

В даному розділі були зазначені технічні вимоги до станції, на якій буде використовуватися інтелектуальна ігрова система з використанням нейронної мережі. Зазначення технічних вимог необхідно для виконання двох цілей – чіткого формулювання характеристик продукту для подальшого використовування даної інформації при продажу системи користувачам, по-друге, для формулювання вимог для тестування або демонстрації системи.

7 ОБРОБКА АВАРІЙНИХ СИТУАЦІЙ

Аварійними ситуаціями можуть бути:

- такі дії користувача в системі, що не відповідають правилам;
- порушення належного функціонування системи;
- збій системного програмного забезпечення (операційної системи);
- відмова апаратного забезпечення комп'ютерів.

При помилках в роботі апаратної складової комп'ютерів відновлення функціонування ІП покладається на операційну систему[7].

При зверненні до Служби технічної підтримки, необхідно точно і грамотно сформулювати питання, які потребують роз'яснення, і описати проблеми, які потребують вирішення. Для оперативного вирішення проблеми рекомендується включати до звернення наступну інформацію:

- точну дату і час виникнення аварійної ситуації;
- докладний опис проблеми, покроковий опис дій, що призвели до виникнення помилки;

Порушення належного функціонування

Порушення належного функціонування системи може бути пов'язано з неправильним настроюванням системи.

Таблиця 7.2 – Аварійні ситуації функціонування

Код аварійної ситуації	Опис аварійної ситуації	Необхідні дії користувача в аварійній ситуації
Помилка 20: Порушення належного функціонування системи	Функціонування системи не відповідає функціонуванню, описаному в експлуатаційній документації	Користувачеві необхідно звернутися до системного адміністратора або в Службу технічної підтримки
Помилка 21: Неправильне відображення або невідображення інформації в системі	У деяких випадках, особливо після оновлення версії системи в кеш-пам'яті браузера може зберігатися застаріла інформація відображення сторінок системи, що може викликати помилки або невідображення актуальної інформації	Перезавантажити систему

Висновки до розділу

В даному розділі було розглянуто можливі аварійні ситуації та способи реагування на них. Інформація з даного розділу може бути доданою до інструкції користувача або до інструкції системного адміністрація робочої машини, на якій встановлено інтелектуальну ігрову систему.

ВИСНОВКИ

В результаті виконання дисертації було проведено аналіз типів штучного інтелекту. На базі цього аналізу сформовано модель інтелектуальної ігрової системи з використанням нейронної мережі . В процесі роботи виконано наступні завдання:

- оглянуто існуючі типи штучного інтелекту;
- методи машинного навчання;
- на базі методу глибокого навчання з підкріпленням розроблено модель;
- створено ігрову систему та в неї імплементовано модель.

Результатами досліджень є модель, на основі якої було створено прототип та проведено його тестування. Для створення була використана технологія Python.

Також у даній роботі було запропоновано узагальнений варіант формалізації у вигляді UML діаграм, що може бути використаний як основа для реалізації аналогічних рішень і продовження подальших досліджень.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Проблеми штучного інтелекту в ігровій індустрії [Електронний ресурс] – режим доступу до ресурсу: http://www.rusnauka.com/pdf/241996.pdf (дата звернення 23.11.2018) – Назва з екрану
2. Генетичні алгоритми [Електронний ресурс] – режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC (дата звернення 5.06.18) – Назва з екрану
3. Bayesian networks [Електронний ресурс] – режим доступу до ресурсу: https://habr.com/post/276355/ (дата звернення 7.09.2018). – Назва з екрану
4. Soft Actor Critic—Deep Reinforcement Learning with Real-World Robots [Електронний ресурс] – режим доступу до ресурсу: https://bair.berkeley.edu/blog/2018/12/14/sac/ (дата звернення 01.12.2018). – Назва з екрану
5. Cluster analysis [Електронний ресурс] – режим доступу до ресурсу: http://statlab.kubsu.ru/sites/project_bank/claster (дата звернення 2.12.2018). – Назва з екрану
6. Gradient descent [Електронний ресурс] – режим доступу до ресурсу: https://towardsdatascience.com/gradient-descent-simply-explained-1d2baa65c757 (дата звернення 1.10.2018). – Назва з екрану
7. Python [Електронний ресурс] – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python (дата звернення 4.10.2018). – Назва з екрану

8. R [Електронний ресурс] – режим доступу до ресурсу: https://www.guru99.com/r-programming-introduction-basics.html (дата звернення 18.09.2018). – Назва з екрану
9. JS [Електронний ресурс] – режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата звернення 2.12.2018). – Назва з екрану
10. C++ [Електронний ресурс] – режим доступу до ресурсу: https://www.geeksforgeeks.org/c-plus-plus/ (дата звернення 1.06.2018). – Назва з екрану
11. Java [Електронний ресурс] – режим доступу до ресурсу: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/jjdev/Java-overview.html#GUID-17B81887-C338-4489-924D-FDDF2468DEA7 (дата звернення 22.05.2018). – Назва з екрану
12. Lisp [Електронний ресурс] – режим доступу до ресурсу: https://blog.ine.com/2010/07/05/a-high-level-overview-of-lisp (дата звернення 1.12.2018). – Назва з екрану
13. OMG Unified Modeling Language [Електронний ресурс] – режим доступу до ресурсу: https://www.omg.org/spec/UML/2.5.1 (дата звернення 31.05.2018). – Назва з екрану. [Електронний ресурс] – режим доступу до ресурсу: (дата звернення 1.12.2018). – Назва з екрану
14. OMG Unified Modeling Language [Електронний ресурс] – режим доступу до ресурсу: https://www.omg.org/spec/UML/2.5.1 (дата звернення 31.05.2018). – Назва з екрану.
15. Nilesh J. Uke, Ravindra C. Thool. Objects tracking in video: a object-oriented approach using Unified Modeling Language.International Journal of Computational Vision and Robotics (IJCVR), Vol. 5, No. 2, 2015.

16. Infantino, I., Cossentino, M. and Chella, A. An agent based multilevel architecture forrobotics vision systems', Proceedings of the International Conference on ArtificialIntelligence, IC-AI Las Vegas, pp.386–390, 2002.
17. Shi-Xiang, T. and Wang, S. The conceptual design and simulation of mechatronic systembase on UML, 2010 2nd International Conference on Computer Engineering andTechnology, pp.V6-188–V6-192, 2010.
18. OMG Unified Modeling Language [Електронний ресурс] – режим доступу до ресурсу: https://www.omg.org/spec/UML/2.5.1 (дата звернення 31.05.2018). – Назва з екрану.
19. Gavrilescu, M. Towards UML software models for cyber physical systemapplications, 2012 20th Telecommunications Forum (TELFOR), No. 2, pp.1701–1704, 2012.
20. Evans, A.S. and Wellings, A.J. UML and the formal development of safety-critical real-time systems, Applicable Modelling, Verification and Analysis Techniques for Real-Time Systems, IEE Colloquium on, Vol. 2, No. 1, pp.2–5, 1999.
21. Yuan, L., Tang, T. and Liu, Y. Research on third-party test bench of train control system using UML, 2011 IEEE International Conference on Service Operations, Logistics, and Informatics (SOLI), pp.560–564, 2011.
22. UML examples and algorithms [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу http://www.omg-portal.ru/articlematerial17 (дата звернення 01.05.2018) – Назва з екрана.
23. SWOT analysis [Електронний ресурс] – режим доступу до ресурсу: http://powerbranding.ru/biznes-analiz/swot/ (дата звернення 1.12.2018). – Назва з екрану

24. Potential channels evaluation [Електронний ресурс] – режим доступу до ресурсу: http://www.businessdictionary.com/definition/potential-market.html (дата звернення 1.12.2018). – Назва з екрану
25. Conquer New Markets in 3 Steps [Електронний ресурс] – режим доступу до ресурсу: https://www.inc.com/minda-zetlin/how-to-conquer-new-markets-in-3-steps.html (дата звернення 1.12.2018). – Назва з екрану

ДОДАТОК А – ДІАГРАМА ПОСЛІДОВНОСТІ

ДОДАТОК Б – СПРОЩЕНА МОДЕЛЬ НЕЙРОННОЇ МЕРЕЖІ

ДОДАТОК В – ДІАГРАМА ПРЕЦЕДЕНТІВ

ДОДАТОК Г – ДІАГРАМА ЗАСТОСУВАНЬ

ДОДАТОК Д – СТРУКТУРНА СХЕМА

ДОДАТОК Е – ДІАГРАМА ДІЯЛЬНОСТІ “ХІД ГРАВЦЯ”

ДОДАТОК Є – ДІАГРАМА ДІЯЛЬНОСТІ “ЗБЕРЕЖЕННЯ СТАНУ НЕЙРОННОЇ МЕРЕЖІ”

ДОДАТОК Ж – ПУБЛІКАЦІЯ

CERTIFIKÁT

MEZINÁRODNÍ VĚDECKO-PRAKTICKÁ KONFERENCE

www.rusnauka.com

MEZINÁRODNÍ VĚDECKO-PRAKTICKÁ KONFERENCE

ZPRÁVY VĚDECKÉ IDEJE

г. Прага

22 - 30 октября 2018

Секция:
Современные информационные технологии

Авторы:
Ліхоузов С.О, Ліхоузова Т.А.

Доклад на тему:
Проблеми штучного інтелекту в ігровій індустрії

MEZINÁRODNÍ VĚDECKO-PRAKTICKÁ KONFERENCE

Председатель оргкомитета Piter Novak

Publishing house Education and Science s.r.o.

CONTENTS

MODERNÍ INFORMAČNÍ TECHNOLOGIE

Výpočetní technika a programování

Software

Zabezpečení informací

Додаток З – Лістинг

```
hid_u = BOARD_SIZE
result_u= BOARD_SIZE

pos = tf.placeholder(tf.float32, shape=(1, BOARD_SIZE))
lbls=           tf.placeholder(tf.int64)
lrn_rate=   tf.placeholder(tf.float32, shape=[])
WWW = tf.Variable(tf.truncated_normal([BOARD_SIZE, hid_u],
             stddev=0.1 / np.sqrt(float(BOARD_SIZE))))
tfvar1 = tf.Variable(tf.zeros([1, hid_u]))
tfvar2 = tf.tanh(tf.matmul(pos, WWW) + tfvar1)
TFVAR3 = tf.Variable(tf.truncated_normal([hid_u, output_units],
             stddev=0.1 / np.sqrt(float(hid_u))))
b2 = tf.Variable(tf.zeros([1, output_units]))
lgts = tf.matmul(tfvar2, TFVAR3) + b2
probs= tf.nn.softmax(lgts)

init = tf.initialize_all_variables()
cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_lgts(
        lgts, labels, name='xentropy')
step = tf.train.GradientDescentOptimizer(
    learning_rate=learning_rate).minimize(cross_entropy)
session = tf.Sessionion()
session.run(init)
TRAINING = True
def play_game(training=TRAINING):
    ship_left = np.random.randint(BOARD_SIZE - SHIP_SIZE + 1)
    ship_positions = set(range(ship_left, ship_left + SHIP_SIZE))
    b_pos_l= []
    actlog= []
    hit_log = []
        current_board = [[-1 for i in range(BOARD_SIZE)]]
    while sum(hit_log) < SHIP_SIZE:
        board_position_log.append([[i for i in current_board[0]]])
        probs = session.run([probabilities],
feed_dict={pos:current_board})[0][0]
        probs = [p * (index not in action_log) for index, p in
enumerate(probs)]
        probs = [p / sum(probs) for p in probs]
        if training == True:
            bomb_index = np.random.choice(BOARD_SIZE, p=probs)
        else:
            bomb_index = np.argmax(probs)
        hit_log.append(1 * (bomb_index in ship_positions))
        current_board[0][bomb_index] = 1 * (bomb_index in ship_positions)
        action_log.append(bomb_index)
    return board_position_log, action_log, hit_log
len_g= []
TRAINING = True
ALPHA = 0.1

for game in range(10000):
    board_position_log, action_log, hit_log = play_game(training=TRAINING)
    game_lengths.append(len(action_log))
    rewards_log = rewards_calculator(hit_log)
    for reward, current_board, action in zip(rewards_log, board_position_log,
action_log):

        if TRAINING:
```

```
            session.run([step],
                feed_dict={pos:current_board, labels:[action],
learning_rate:ALPHA * reward})
```